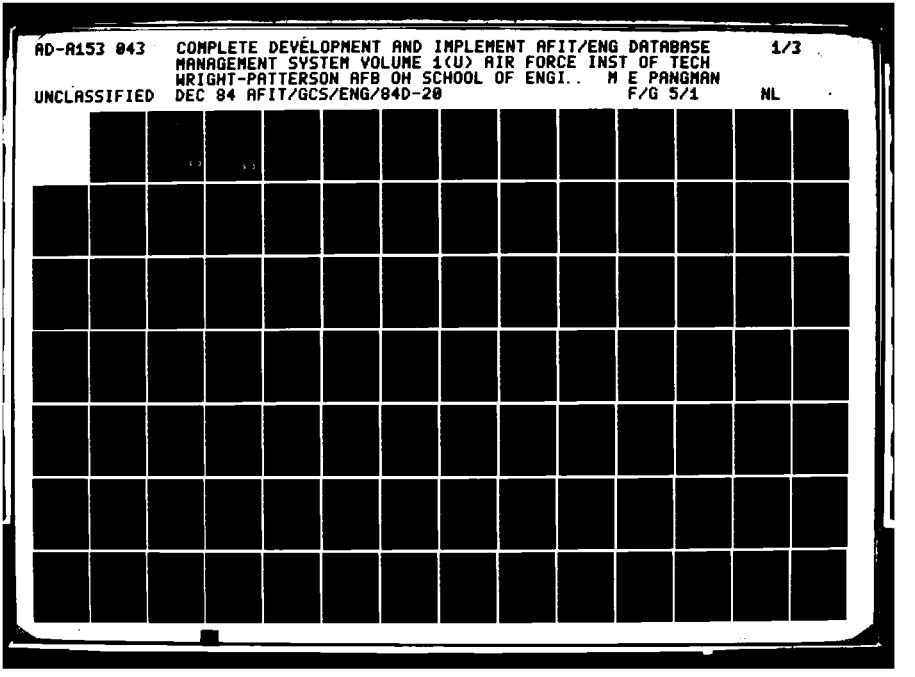
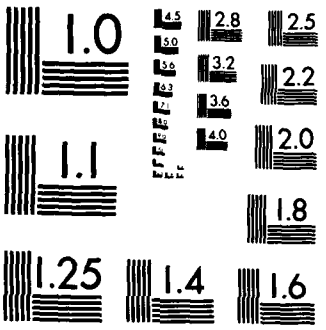

AD-A153 043 COMPLETE DEVELOPMENT AND IMPLEMENT AFIT/ENG DATABASE MANAGEMENT SYSTEM VOLUME 1(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. M E PANGMAN
1/3
UNCLASSIFIED DEC 84 AFIT/GCS/ENG/84D-20 F/G 5/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

REPRODUCED AT GOVERNMENT EXPENSE

DTIC

AD-A153 043

COMPLETE DEVELOPMENT AND IMPLEMENT
AFIT/ENG
DATABASE MANAGEMENT SYSTEM
Volume I

THESIS

Myron E. Pangman
Major, USA

AFIT/GCS/ENG/84D-20

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DTIC
ELECTE
MAY 1 1985
S D
B

DTIC FILE COPY

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

85 4 05 031

COMPLETE DEVELOPMENT AND IMPLEMENT
AFIT/ENG
DATABASE MANAGEMENT SYSTEM
Volume I

THESIS

Myron E. Pangman
Major, USA

AFIT/GCS/ENG/84D-20

DTIC
ELECTE
S MAY 1 1985 D
B

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

AFIT/GCS/ENG/84D-20

COMPLETE DEVELOPMENT AND IMPLEMENT AFIT/ENG DATABASE MANAGEMENT SYSTEM

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Myron E. Pangman

Major, USA

December 1984

Approved for public release; distribution unlimited

Preface

The AFIT/ENG database contained the means to store student data and the ability to manipulate this data utilizing the TOTAL Database Management System. This study was undertaken to expand the database to include faculty data and additional data-elements for use in future applications, and to provide a standard means of database input and output utilizing the Forms Management System (a software form presentation utility), which would show standard forms to the user on the CRT for the input and output of data to the database, and to enhance the database capabilities.

I thank my advisor, Dr. Gary Lamont, for providing a thesis topic which was most interesting, instructional, and a useful tool. I also thank him and the members of my thesis committee for their directions and help. Additionally, I thank Robert Ewing for his support and many hours of expert assistance. I thank my classmates for their much appreciated assistance and the Army for providing the opportunity to attend this course of instruction. Finally, I acknowledge my deep gratitude to my wife, Kerri, for her unfailing support and encouragement throughout the term of this in-CONUS short tour.

DTIC COPY INSPECTED

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

Table of Contents

List of Figures

List of Tables

Abstract

This study undertook a complete revision and redesign of the AFIT/ENG database in an effort to expand existing capabilities and utilities. The Software Development Life Cycle approach was utilized throughout this project in order to provide a basis for sound software engineering principles and provide for a sound database development base. The initial compilation of data was obtained by interviewing a wide range of individuals to determine required functions as well as requested data items. The results were combined to form the specific network master and variable data-sets for use with anticipated application programs for a greater database utility.

The expanded database exhibits the means to manipulate student and faculty data as well as provide for the inclusion of advanced data handling routines like education plan and graduate credit record utilities. To insure standard data input and output as well as provide a more user friendly environment, a menu form display technique was incorporated within the expanded database driver. The resultant program was designed for modularity and expansion with little additional coding required to connect future database application routines.

Chapter 1

I. Introduction

Background

The means and methods for storing data in a computer based record keeping system has greatly expanded from the initial development of the computer (1:3). Scientists and academicians have developed and improved many methods to the point that one must have knowledge of a number of database management systems in order to choose effectively among them when determining which his organization will use. The four major components of any database system are contained in data, hardware, software and users (1:4).

The Air University, as with any organization, requires that certain data be passed among its different functional units. As with the nature of individuals, it is sometimes easier and more time efficient to look up data already available, than to seek anew the answer each time it is asked. A prototype database has been established in the Department of Electrical Engineering Digital Engineering Laboratory in an effort to provide maximum classroom instruction for its students and at the same [illegible]sist the School of Engineering.

[illegible] in his 1980 thesis (2), proposed the establishment of a Consolidated AFIT Database. A database 'test-bed', under the direction of Dr. Gary Lamont, Department of Electrical Engineering, AFIT, was constructed in the Fall of 1983 and located in the Electrical Engineering Digital Engineering Laboratory (now the Informational Sciences

The resultant subject categories centered primarily around improving the existing portion of the AFIT/ENG Database:

a. User Interface
    1. Revising default actions
    2. Improving menus
    3. Improving user-friendliness
b. Design/implementation
    1. Adding additional links
    2. Expanding the context of the data contained with the database
c. Adding application programs to assist class and student advisors and Administrative efforts

This thesis effort was initiated utilizing these subject responses but were not limited to them.

Data Base Concepts

The interview results were analyzed to determine how data should be structured and related, and which items should be included in the database utilizing a network DBMS. Such an aspect of database system implementation involves translation of the specification for the database architecture, which is formulated independently of any DBMS, into the characteristics required by database definition under a particular DBMS. This is accomplished using a collection of statements in the selected and implemented DBMS package data description language.

Operations on the database itself are invoked by application programs. These application programs may be run as jobs in the jobstream or may operate on transactions received from a communications interface.

'corporate memory', those interviewed were most cooperative and encouraging. All completed interviews were later transcribed to disk to assist in compilation and determine requirements. Responses were, for the most part, confined to the topic at hand, yet a number of subject areas readily appeared. The subject's individual responses to the interview became extensive, and are contained in Appendix B.

Subject Topic Responses

Upon closer scrutiny of the interview results, a number of more specific topics were determined to accurately cover the subject areas of interest. Table 2 contains a list of major topic areas uncovered during the interviews, with the number of interviewees who requested data or application programs pertaining to the listed category. These categories were provided by the subjects in order to help themselves keep track of their responses during the interview. The categories listed reflect a reduction as some data spanned more than one category. The contents of each category are found in Appendix B.

Table II

Major Subject Areas

CATEGORY	NUMBER OF TIMES SELECTED
Class Advisor	7
Management Tool	6
Data Requirements	8
Scheduling	3
Student Information	8
Security	5
User Friendliness	8
Faculty Information	6

prior to conducting the personal interviews, each subject received a copy of the data questionnaire to familiarize themselves with both the content and purpose of the interview. Appendix A contains the questionnaire.

The questionnaire outlined the purpose of the interview, introduced the interviewer, explained the use of the interview results, and encouraged subject responses spontaneity in order not to limit the scope of the interview and encourage the subject to feel free to discuss any areas of the present or proposed database configuration. The individuals interviewed ranged from Department Head, Deputy Department Head, current and future Class Advisors, instructors and secretaries who utilize the AFIT/ENG Database. Table 1 shows the range of subjects interviewed.

Table 1

Interview Mix by Category

CATEGORY	NUMBER
Class Advisor	6
Secretaries	2
Professors/Instructors	10
Department Administration	2
Students	5

The questionnaire is self-explanatory in nature, intended to act only as a means of eliciting comments and to thinly confine the topic matter. To insure complete and accurate results, and help the subject's to feel more at ease, all interviews, with one exception, were cassette-taped and conducted in the subject's office. Interviews ranged from 20-90 minutes in length. Although this was another series of interviews in conjunction with a thesis investigation regarding the AFIT database, perhaps due to the quick personnel turnover and subsequent short

implemented in this manner, it should be able to expand, if necessary, to encompass additional Electrical Engineering Department requirements and other departmental areas within the AFIT environment.

The interviews were obtained from present and most likely users of the Database Management System (DBMS). Because of current design constraints, present users are few and mostly confined to administration and class and student advisors. Most likely new users were defined first to be incoming class advisors with academic advisors (most faculty members) to follow second. Although no final implementation plan was determined, individual student use of the database (apart from entering initial personal data and individual education plan) is anticipated as a later expansion (Appendix N). The interview data results were useful in assisting the selection of the best potential data items for inclusion in the database, projected as far into the future as can be accurately and practically planned (7:68).

Interview Methodology

An important requirement was to determine exactly what information was needed to plan the database development. Previous interviews accomplished by Allred (2:20) and Ricks and Colburn (3:IIIA.i) were conducted to determine specific data items for inclusion in a database. The methodology used during this thesis effort was similar.

The manner in which an interview is conducted can affect the type of data collected and subsequently made regarding the implementation scheme. Therefore, rather than obtaining only administrative input requests or only asking respondents to fill out an interview form, a more coordinated and free flowing personal interview was decided upon. At least a day

This was accomplished by reviewing a number of official and unofficial communications:

1. By studying the past requested data requirements;
2. By reviewing data generated within, as well as data directed into, the Electrical Engineering Department;
3. By obtaining copies of data kept by both the Electrical Engineering Department and those individuals who work within it.

To shorten this process, personal interviews of individuals who use and will most likely be using the database were conducted. The results were captured and collated to relate to functional activities. This information provided a means of identifying data item requirements as well as user requests for data included in the database (7:21).

Identifying data item requirements is a major step to determine the direction of subsequent work on this staged development database thesis. The previous thesis efforts directed at determining the composition of such a database must be screened and results weighed to insure the chosen direction for the database is carefully considered. What is not desired is a database placed in the hands of individual designers and programmers who implement applications on a function-by-function basis using the package simply as an access method. The result is an inevitable replication of the old file-oriented systems, only with more keys than before and with more computing resources consumed (7:1). Centralized control demands a centralized description of both the problem environment and the supporting data. It also requires centralized control over the design, implementation and maintenance of the architectural structures that make the database system useful for the user community (7:1). With the system

Chapter 2

II. General Approach to Requirements and Specifications

Previous work was done by Allred (2) and Ricks and Colburn (3) to propose and define an AFIT database. These studies envisioned the whole of the Air University system tied into one database encompassing each AFIT school, administrative requirements, as well as related services (bookstore, etc.). This chapter covers the techniques used to obtain the data for the analysis used in the Software Development Life Cycle.

A database 'test-bed', under the direction of Dr. Gary Lamont, Department of Electrical Engineering, AFIT, was constructed in the Fall of 1983 and located in the School of Engineering Digital Engineering Lab (now the IS Laboratory). This was accomplished to acknowledge the need for an AFIT database. Initially, the database was restricted to contain only data relating students and their selected courses.

This thesis effort is to expand the initial design and implement additional features to allow it to be more responsive and useful for day-to-day activities within the AFIT Electrical Engineering Department. Some remaining requirements which exist are to determine what further system design is needed, what priorities should be placed on the additional design items, and how to best merge these additional requirements with the already operational database.

Organizational Requirements

With the user's need already established, the initial step was to define the AFIT/ENG Department of Electrical Engineering organizational requirements to determine what data should be kept within the database.

development and implementation of the project is neither expected nor desired. The uniqueness of the selected network Database Management System is explained on a need-to-know basis in order to adequately assist the reader.

Material and Equipment

Implementation and testing utilized the existing TOTAL Database Management System, VAX 11/780 computer with the VMS operating system, and application programs. An additional mini database was established with copies of existing application programs to accomplish testing and debugging. In order to further isolate this project from the present database, sufficient disk memory was on hand to insure a large enough area to hold and manipulate the revised database separately from the present AFITDB.

the results, and to take advantage of representing movement and transformation without regard to the passage of time during this view.

Structure Charts (Appendix M) provide a visible and convenient method for portraying the interrelationships between the individual software modules. Hierarchical and scope of control relationships can be easily seen, and parameter passing, in the form of data and control flags, can be effectively identified between modules.

Data Flow Diagrams (Appendix M) are a graphic depiction of the system specification that identifies inputs, desired outputs, and data transformations. This usually leads to the definition and depiction of modules and their relationship to one another and to various system elements, in a form called a structure chart.

Further Development Aids and Assumptions

To insure non-interference with the current in-use database, a mini database was constructued for testing purposes with the same logical characteristics (file names, database descriptor names, elements) as the live database, but containing only a subset of its data (i.e., 5% - 10%) (4:D-21). After validating that an application program utilizing the present database was valid, the data was transfered into the expanded database to insure compatibility. Application programs were designed, written, debugged and run through the mini database first, prior to utilizing the complete database. After this the completed application programs will be included in the expanded database.

It is assumed that the reader has a good technical background and is somewhat familiar with database concepts. It is further assumed that continuous and exhaustive explanations for each path chosen during the

message and may return some fixed test value. The stub is eventually replaced by the full module which would then include calls to other stubs. In this manner, an entire system can be gradually developed and tested (4:212).

A major development in facilitating the programming task is known as 'structured programming'. This premise is to use a small set of simple control and data structures. A program then is built by nesting these statements inside each other. This method restricts the number of connections between program parts and thereby improves the comprehensibility and reliability of the program. The 'if-then-else', 'while-do', and 'sequence' statements are one commonly suggested set of control statements for this type of programming (4:211).

Development Tools and Documentation Aids

Two Development Tools and Documentation aids were chosen from among the many currently available. Selection criteria included:

1. Clarity of presentation
2. User familiarity
3. Ease of modification
4. Availability of automated storage and retrieval
5. AFIT requirements

(5:15)

Structure Charts were chosen over Structured Analysis and Design Technique (SADT) diagrams because they were rated higher in 'module communication', 'training need', and 'proliferation' (5:68). Likewise, Data Flow Diagrams were chosen as an antithesis to the top-down approach in order to reduce the effect of the designer's experience and biases on

The testing stage may require up to half the total development effort. Testing is divided into three distinct operations:

1. Module Testing - tests each module against the test data supplied by the programmer
2. Integration Testing - tests groups of components together
3. Systems Testing - tests the completed system by an outside group

(4:200)

Additional types of testing encompasses boundary value testing, path testing, equivalence partitioning, static testing, etc. However, the above three listed operations adequately cover the useful range of testing used within this project.

Software Engineering Techniques

The Software Engineering Techniques used in developing the software system included Top-down Design, Top-down Development, and Structured Programming. Top-down Design is a technique in which a programmer first formulates a subroutine as a single statement, which is then expanded into one or two of the basic control structures (SADT, Data Dictionary, Data Flow Diagram, etc.). At each level, the function is expanded in increasingly greater detail until the resulting description becomes the actual source language program. Using this approach, also called 'stepwise refinement', the program is hierarchically structured and is described by successive refinements (4:211).

Top-down development is another technique for implementing hierarchically structured programs. Here, the top-level routines are written first, and the lower level routines, called stubs, are written to interface with these. The stubs return control after printing a simple

1. Requirements Analysis
2. Specification
3. Design
4. Coding
5. Testing
6. Operation and Maintenance

(4:198)

The focus of the Requirements Analysis rests on the interface between the computer, used as a tool to solve some problem, and the people who need to use it. A Requirements Analysis can aid in understanding both the problem and the tradeoffs among conflicting constraints, thereby contributing to the best solution (4:199).

While Requirements Analysis seeks to determine whether to use a computer, Specification seeks to define precisely what the computer is to do, but not how to do it. Issues that are examined at this stage include input and output record formats, database layouts, algorithm selections, etc. (4:199).

In the Design stage, the algorithms called for in the Specifications are developed, and the overall structure of the computer system takes shape. The system is divided into small parts (modules, with constraints as to function, size, and speed (4:200).

Coding is the stage which includes the use of high-level languages and structured programming in order to produce the mechanism to solve the task at hand. Since it appears that most errors found in a project occur in the design stage, the coding stage apparently has been mastered better than any other (4:200).

Approach

The project began with preliminary literature research on those theses which led to the present AFIT/ENG Database. Personal interviews of current and possible users of this database were conducted to gather information with which to support the expanded database. Because of the time (four years) from the initial concept of Allred's AFIT Database, and the partial implementation of the current database, determining updated information as to user likes, dislikes and requirements was necessary. Not surveying this information may accept blind implementation of specifics proposed for different requirements. Therefore, after validating the database design requiremnts and structuring the design, the proposed application programs were began.

The contents of the current AFIT/ENG database and application programs were determined and cataloged in a configuration control document to provide a departure point from which to build the remainder of the structure. The Software Development Life Cycle was followed as a guide to develop, operate and maintain the system. The Software Engineering techniques used in this effort can be classified as 'top-down' and 'structured'. Both of these techniques have proven useful (6) during the Software Development Life Cycle of a project.

Software Development Life Cycle

To help control the development of a project, six separate stages have been identified through which software projects must pass. These six stages make up the Software Development Life Cycle:

existing student data, incorporating a unique form display technique, and including provisions in the schema and database for proposed application programs.

Scope

The main effort of this project is to complete development and implement the modifications of the AFIT/ENG database with specific use for the Department of Electrical Engineering. Unlike previous theses, this effort is confined to acquiring and using student, instructor and course data utilized within the AFIT/ENG administrative arena. Although recognizing the need for an all-inclusive AFIT and Air University database (2), such a total effort is not now supportable. Thus, no specific work is accomplished toward implementing the CADIS design (3) for the rest of the Engineering School, Logistics School, AFIT Administration, scheduling, operations, personnel, or Air University. This effort is believed to not only provide additional insight and feedback into the future development of such a database, but also provide immediate use of specific application programs not now available.

Current Knowledge

The AFIT/ENG Database currently encompasses most needed student data. Application programs exist which produce individual course enrollments, reduced individual student Education Plans and class and course rosters (see DBTA, Informational Sciences Laboratory). At this stage the database is 'student' and 'course' oriented.

Laboratory). Emphasizing the aspects of data security, academic environment, and economic reality, this database has been installed on VAX/VMS 11/780 which has a limited number of users and terminals. The database itself is limited in scope and incomplete in structure. Dr. Lamont's overall objective is to establish a working database useful within the Department of Electrical Engineering, AFIT.

As happens within organizations, each Department stores information dealing with assigned students and faculty which is duplicated not only within itself but also the administrative areas. The proposal of the Consolidated AFIT Database was an attempt to reduce this data redundancy, yet provide accurate data quickly when required. The inclusion of each student's course listing by quarter was introduced to alleviate the duplication of requiring every student to register each quarter thus reducing scheduling delays as well as confusion between departments. This Consolidated Database also acts to reduce redundant application programming to increase overall efficiency and productivity.

This proposal was followed by another database thesis proposal (3), classroom projects in EE 646 (Computer Database Systems), and DBTA efforts. The opportunity for students to work within a database and design, write, and use schemas and application programs while studying these concepts allows maximum use of these available resources.

Statement of Problem

The purpose of this thesis investigation is to complete development and implement the AFIT/ENG Database utilizing the TOTAL Database Management System. Proposed implementation encompasses required faculty data schema entries, additional academic related schema entries, expanding

Whatever the source of activity, the application program executes a logically constructed sequence of call statements on the DBMS. Each of these calls makes reference to the database under the particular database architecture and for the specific DBMS. A number of basic application program topics (2:21) were determined which mostly encompass the present expansion effort.

Within the database design methodology each existing record's attributes was compared with those determined to satisfy the proposed database requirements utilizing third normal form attribute techniques. Third normal form can be loosely defined as a relation 'R' with a family of functional dependencies that have no transitive dependencies (8:236). This is useful for the following reasons:

a. Avoids redundancy

b. Forces close scrutiny of attributes

c. Assists in data independence

Appendix C contains the data entities and information determined to be needed to expand the current AFIT/ENG Database. This evolved from the data obtained from the subject's responses outlined in Appendix B, the data items contained in the original AFITDB (Appendix E), and the individual record attributes list (2:Appendix B). Each area cited for revision, inclusion, or improvement was examined to obtain those items which would satisfy the expected database requirements. Although the original attributes were determined utilizing a relational database design, and this effort is designed utilizing the TOTAL network DBMS, similarities in data items and files (relations) result. Normalizing relations in the relational approach assist in selecting those data-set

items for inclusion within a network data-set, since the items relate to their atomic values (1:64-78). Direct comparisons cannot in all cases be made.

Prioritizing New Database Requirements

A more subjective determination of which interview responses should be implemented and in what priority was made by prioritizing the finalized list of interview responses as contained in Appendix D. This analysis was a DBMS coordination exercise with the user (7:39). It was obvious that some items did not belong in what is envisioned as a school support multi-user Database (such as keeping track of the paperwork required to acquire additional computer resources, and maintaining mailing lists). It was also clear that all items mentioned could not be accomplished in one project. Table III contains the expanded subject response area investigated for further work within this project. Upon closer scrutiny, it was decided that portions of items 4 and 15 were adequately covered in other records (existing or planned) and were deemed inappropriate for inclusions within this database. They were eliminated from further consideration.

Table III

Expanded Subject Responses

NO.	PRIORITY	EXPANDED STUDENT RESPONSES
1	*	Revise Default Thesis Course Credit
2	*	Improve Menus in Database
3	*	Additional Links
4		Internal Course Data
5	*	Include Army Short Course Students in Database Calculations
6	*	Link Instructors to Courses
7	*	Student Education Plan
8	*	Determine Teaching Loads
9	*	Graduate Record Data
10	*	Instructor Statistics
11	**	Room-to-Course Scheduling
12	*	Security/Integrity
13	***	Course Grading
14	**	Faculty Professional Duties
15		Tracking the Acquisition of Computer Equipment

The remaining items were awarded a three tiered status:

1. First priority - *
2. Second priority - **
3. Third priority - ***

Ten items obtained top priority (1-3, 5-10, 12), two items obtained second priority (11, 14), and one item obtained third priority (13).

The highest priority was awarded to those items which expand the existing database, complete functions usually considered lengthy and repetitive (prime candidates for computer support), and which currently have no other means for accomplishment other than by hand. This was accomplished by creating additional links to existing data elements with newly created data elements in an effort to more efficiently utilize and easily expand the current database. Included in this expansion are

proposed utilities to assist the Administration, Faculty, and Students to determine mid-course and end-of-course requirements.

An example of this is the MDEG master data-set. This data-set contains data elements consisting of the degree name and its corresponding designated departmental number with links to individual courses and course sequences. Extrapolating this into an as yet to be written application program will provide a utility to check a student's degree or separate sequence qualifications.

Some requested utilities, although seemingly useful, were not determined to be best suited for inclusions at this time. A few subjects requested a room-to-course scheduling utility (item 11, Table III) which ENA presently determines manually. However, until the database is able to include most enrolled AFIT students, a separate utility for this purpose will fail to be adequate (although the anticipated data requirements were included in the database schema). Also, some instructors requested a facility for determining course grading data. This also was not awarded a high priority due in part to the varying grading methodologies and the availability of a grading program available on many of the microcomputers within the Department.

Constraint Specifications

A complete database restructuring would utilize previous data to insure that selected hardware and software would provide the sought after results of a well defined database with ample recall and use utilities. Inasmuch as the VAX and TOTAL DBMS were among those items deemed capable of supporting the Department database requirements, this selection

alternative was not considered. The selection of a network over a relational or hierarchical scheme was centered around the ease of setup (network schema vs relational third normal form or better schema requirements), use, and availability of DBMS.

This thesis investigation was undertaken to expand the current database utilizing existing resources. Some constraint specifications and required resources used to accomplish this project encompass the following:

- VAX 11/780 in Electrical Engineering Digital Engineering Laboratory
- TOTAL DBMS installed on this VAX dictates the method in which applications programs must be written
- Utilize established host language - PASCAL
- Use the VAX-hosted Forms Management System to produce and call menus and input/output data (to reduce screen clutter when providing help to users during input/output operations and also to reduce amounts of required code within application programs)
- Access to the current AFIT/ENG Database
- Access to the current database application programs
- Establish a Mini-database for testing and debugging applications programs
- Ample memory to accommodate the above specifications and resources

The specification of abstract data types by the PASCAL high-order programming language lends versatility for database programming since it allows the programmer to define distinct data types and structures. Also, since PASCAL is used and taught within AFIT, PASCAL was selected for use in writing the code for this thesis effort. In this way, database users

and administrators would already have some familiarity with the database language used for future database examinations and/or modifications.

The input and output of data to and from the database has been routinely handled by extensive and verbose prompting requiring many lines of separate code, for input or output data. As an alternative to this, incorporating the use of the utility, Form Management System (see Chapter 4), provided an easily transportable and modular form capability which is capable of more easily altering each form or field within a form than the present system which utilizes embedded code within the application program. Additionally, close contact with the Database Management Administrator (DBMA) and Database Technical Advisor (DBTA) were most useful to maintain maximum support and eliminate student confusion. Maintaining data independence and security of the database were also prime concerns for these insure that the database can be transferred if necessary and data contained within the database is relatively safe from unscrupulous means.

The life-cycle approach for the development of the AFIT database includes more specific determinations for the proposed contents of the database. Additional application programs are expected to result following such reviews. During the database life cycle, extensive performance appraisals of both the data relationships and programs within the database must continue to be conducted.

Throughout the construction life of this staged database, Security/ Integrity procedures must continue to also be developed and investigated. As the type and number of users increases, software and procedural security procedures for such aspects as what data users may access and how

they may utilize it will become more important (9:193-194). Unauthorized acquisition and modification of data must be an integral consideration when adding additional application modules and programs (10:42-43).

Summary

This chapter explored the general guidelines surrounding the process of expanding and improving the present Electrical Engineering Department network database management system. Utilizing the Software Development Life Cycle, the user need was determined, and system and software requirements were briefly explained, mostly through personal interviews and previously completed thesis work. The requirements definition was structured in the form of the expanded database DBG (Database Generation Module) which clarified the requirements in order to determine how to best meet user needs. Adapting those needs to the identified constraints, enabled a skillful use of available tools to enhance the DBMS user operation.

Chapter 3

III. Database Design

This chapter will cover the general areas in the design of a database. This stems from the guideline of the software development cycle, requirements definition analysis, and the more specific aspects relating to the implementation of a database. Also, the construction of a database utilizing a three-tiered approach will be discussed and weighed as to a system's success or failure.

Database is the organized collection of data items, real or assumed, stored in a data structure and used to generate the information employed by management in making decisions (11:1;12:20). This is often managed by a Data Base Management System (DBMS), a blend of hardware and software which has custody of the database. The DBMS requires a data base definition which tells it about the data items and data structure that it administers. Database design is a process that leads eventually to such a database definition. Database design produces preliminary output called a logical design, and a second step (database engineering) converts this to the database definition that the DBMS expects.

The data (facts) describe the enterprise or business or school. The information is produced in order to interpret, correlate, measure and express the state of the enterprise. If an enterprise (school) decides to change its accounting practice, procedure, reporting period, or format, it is the procedure, timing, or format of information that changes, not the data (12:p.20).

During data base design ('logical' design) the designer has little to go on but assurances that there are data base 'requirements'. The designer tasks are to isolate the real requirements, gain a view of future requirements and a plan for database evolution, and analyze the database problem so far as to formulate a 'logical' solution. This early phase of design is a time when:

- There is no design.
- Requirements are duplicated, contradictory of ill-formed.
- Fundamental key items are ambiguously defined.
- Expectations of database utility are considerable.
- The existing system is incomprehensible.
- Fantastic (or no) records are available on how the system works.

Here, although a database existed, the limited nature and urgency with which it was initially constructed prompted the task to proceed in the absence of an organized framework. The database design process may be conducted by proceeding through a number of stages:

- Focus on the logical design problem, separating out physical implementation problems (discussed later).
- 'Factor' the data problem, looking for atomic entities and atomic relationships between them.
- Sketch data structures that meet the data problem; if necessary, leave some entities and some relationships undetermined while the structure is being developed.
- Document all the data elements and the relationships among them; use this process to refine the factoring and structuring done earlier.

- On or near completion of the design, review the relationships from a dynamic (additions and deletions) point of view; pursue this review to include changes that propagate through the data structure.

Clearly, the actions taken in the last few steps will often reflect back (and cause changes) to the earlier work. Thus, the process is viewed as proceeding by successive refinements until the logical design is complete (11:2).

Although 'steps' and 'stages' are mentioned in this project, it is emphasized that throughout the process, revision and redefinition are ongoing. There are numerous reasons for this:

1. Some based on previously defined data.
2. Some based on newly included data.
3. Some based on deleted items, which therefore caused either 1 or 2 (above) or redefinition of data or links to take place.

A Three-Tiered Database Approach

Database design is the process of specifying the 'logical' relationships that data items bear to one another, given that these data items are to be maintained in a databased system. In the data processing literature, the work 'logical' is generally another way of saying 'abstract', but it is not true that database design is entirely abstract. The eventual outcome of database design is a working database, which is a highly practical objective (11:4).

A three-tiered approach is a simple way of looking at how a database is constructed. The initial work is that of the logical design. Second is that of the DBMS selection, while part three is the database engineering.

The subject matter of database design is dominated by data. The designer is concerned with that portion of system development that deals with endless details about data: the names of data, the dynamic properties of data, the sources and end uses of data, etc. In spite of these details, the database designer must retain a perspective, in order not to emerge from the database design process with a dull, complicated web of dreary detail. The assignment is to find useful order and place the details into an organizing framework. The result is a structure that depicts what the database will be.

Failure to proceed in this (or like) manner will cause confusion, wasted space in the database and increase the cost in terms of error checking, system response and usefulness. Using a relational approach, an alternate method for database design could be to start with an entity and draw it as a bubble chart depicting dependencies between each attribute. As each attribute is added to the chart, each is checked for similarity to previous data items, redundancy, relationships to other data items and normalization (13:Chapter 15).

Logical Design vs Physical Design

The minimum statement of what the database must contain (logical design) was determined using the following properties:

- The data items supported in the data structures of the database were identified and characterized. This is a data inventory, complete with definitions of data item usage.
- Certain data items were given special attention. These are the items that have a central role in the indexing, linking, keying and naming processes which give the database its organization.

- Numerous rules were specified. These are the rules under which the key data items are connected together in the framework.
- Optinally a few important assumptions were recorded. These concern the data management capability required, the hardware apparatus required and/or the historical records to be preserved.

'Logical database design' is separated with equal ease from the database engineering work which follows. The Logical database design produces concept, framework and related documentation on which the database will be. The Engineering stage yields the database definition (how it will be implemented). Creative elements are added during the Engineering stage like:

- Data item size; details of the form in which individual data items are stored.
- How to deal with variable length records.
- How to deal with variable numbers of data items.
- Selecting a database management system.
- Selecting space management options.
- Employing security techniques. (11:7)

In general, quantification of record sizes, file sizes, etc., is all physical design implementation. The fact that each person is identified by a name and a Social Security Number (SSN) is logical design implementation. Alternatively, access could be determined by an individual's name or class. The fact that access must then be supplied by Social Security Number, or name or class number is also logical, and that access is provided by hashing the SSN (name or class number), or inverting it, or chaining them, is a physical (Engineering) decision.

The logical design keys each person in a database by the selected key. There are also other portions of the database that can be linked to each individual utilizing this key as a 'link'. But is is left to the Database Engineer to decide how that linkage from each person to those other pieces of data will be implemented. The particular technique (or DBMS) used will determine the different methods of achieving identical results.

Utilizing a technique much like factoring, each item of data proposed for inclusion into the database was simplified and clarified into atomic elements. These elements were then grouped according to associations provided in the interviews in order to determine data sets or groups of like data. Those items with an affinity for more than one group were looked at to determine if a separate group would better suit the database needs or whether this might add the associated problem of unnecessary duplication.

From a dynamic point of view, Database can be reflected in four types of computations:

1. Building of the data collection.
2. Updating of data elements in the data collection.
3. Retrieval of data from the data collection.
4. Reduction of large quantities of data to usable form.

(14:5)

AFIT Enginering Database Development

The building of any data processing system is a complex and timely process. There are many variables to be properly managed. If not handled properly, the results could lead to eventual failure of the system. Eventual failure can be defined to be the user's needs not met.

This occurs if the system does not or cannot provide the information the user needs. The system may be unacceptable if the response time is poor. Another design failure example is when a new user requirement (or an old, forgotten user requirement) renders the system obsolete. Other reasons why a system could be considered a failure are if the user's on-line availability requirements cannot be met by application operations or because of failure to fulfill the system's off-line requirements.

The best insurance against such failures is to begin with a well-planned complete system design. However, managing the complex variables involved in building a database application is no easy task. Good database and application design techniques gained from experience, classes, books, and so on, must be merged with the reality of specific user requirements (15:xi).

Selecting a DBMS

One of the aforementioned three major tasks in the construction of a database, DBMS selection, was not dealt with in this thesis, since a network DBMS had already been selected and installed on the host hardware. The type of computer system was also not a part of this project since this also had been procured and is used to support other research work as well as maintain administrative data for the Electrical Engineering Department of the School of Engineering, AFIT.

Since the selection of a DBMS and hardware support system had previously been accomplished, these aspects of constructing a database were thus unnecessary toward the successful completion of the project. This takes some of the physical hardware objectivity out of the task, but this

2500 variable-entry files and vice versa (18:3-5).

Indexing Capabilities

TOTAL relies on calculated indexing (i.e., randomizing) to retrieve data. Single-entry records are located based upon the randomized value of the master key. Variable-entry files may be read serially, although keys are usually accessed through linkage paths for the single-entry file(s). TOTAL allows a single-entry file to be linked to any variable-entry file in the database. In a sense, single-entry files serve as tabular indexes to variable-entry files; queries generally cannot be resolved within single-entry files. In TOTAL, a tabulation represented by a single-entry file may represent a tabular indexing to a number of variable-entry files in the database system. Furthermore, variable-entry files may have such an indexed reference from any member of single-entry files. Users can reference a master file, search through an associated detail file, locate records in that file on certain criteria from which the key values for referencing another master file can be achieved.

Data Access, Entry, Update, and Deletion

Data sets are accessed through a Data Manipulation Language (DML) that is set up in a CALL statement via a host language like COBOL, FORTRAN, PASCAL or ASSEMBLER. Single-entry data files are retrieved directly through a randomization algorithm of its key. Variable-entry files are accessed along a linkage path with a specific master record. Records can be read forward or in reverse (i.e., starting from either

data for processing. It may be described as a partially inverted system organized into a network of file structures. The inverted list is distributed as linkages, chains, or pointers within the data records themselves. The records are all fixed length and are categorized by the type of data file to which they belong. TOTAL can manage virtually an 'unlimited' number of files in an 'integrated,nonredundant' basis and provides for association of each of these files with other related files to form an integrated database.

File Types

There are basically two types of data files, single-entry files and variable-entry files. Records in a single-entry file, which is often viewed as a master file in the traditional file-oriented system, contain the master key for a cohesive information set that is distributed through the associated variable-entry files. The master key can be up to 256 bytes long and must have a unique value. The records themselves are positioned randomly according to a user-transparent randomization of the master key value.

Variable-entry (or detail) files, on the other hand, are organized serially so that successive records are physically stored in the first available data area within a file's allocated space. They are logically linked to other similar data entries in the variable-entry file as well as to the associated single-entry record to which it belongs.

Database Growth

The maximum number of records in a TOTAL database is limited by the peripheral storage available. Any single-entry file may be linked to

deleted. The top-level relations cannot be processed serially because of the use of direct file access; only unordered sequential access is possible. This may force some relations to the bottom level which otherwise would fit better on top.

Since no new relations or new linkage types can be dynamically created, an associative relation, such as supplier_assembly, has to be created at the time the schema is defined and will be updated as its owners change.

TOTAL Design Principles

Opportunities to enhance the basic design should be explored by the TOTAL user. The limitations of TOTAL must also be recognized. With this knowledge of TOTAL, the designer can examine the system needs and identify a practical database design which utilizes the advantages to the fullest and bypasses the drawbacks (16:3-9). The designer must be alert to the nature of the system taking care not to make use of an opportunity just because it exists when the advantage would be only 'icing on the cake'. By the same token, the designer should not be put off by the limitations of the DBMS until the analysis shows that the limitations are an obstacle in this system. The appropriate steps can be taken to resolve the issue. In other words, the database designer should not create a problem where none exists.

TOTAL Characteristics

The TOTAL database management system provides an effective means for organizing and managing diverse data so that application analysts/ programmers can efficiently and conveniently maintain and retrieve the

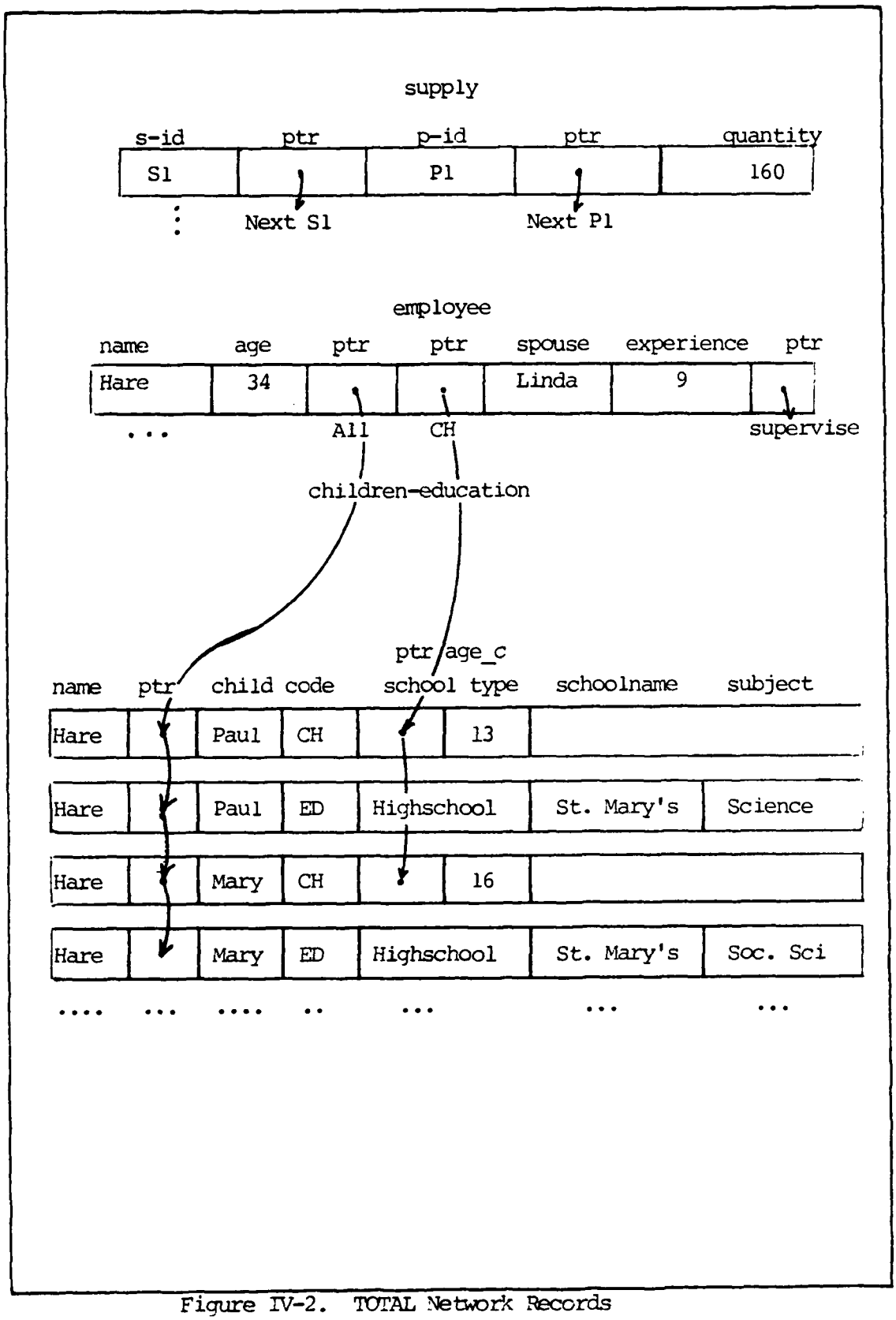


Figure IV-2. TOTAL Network Records

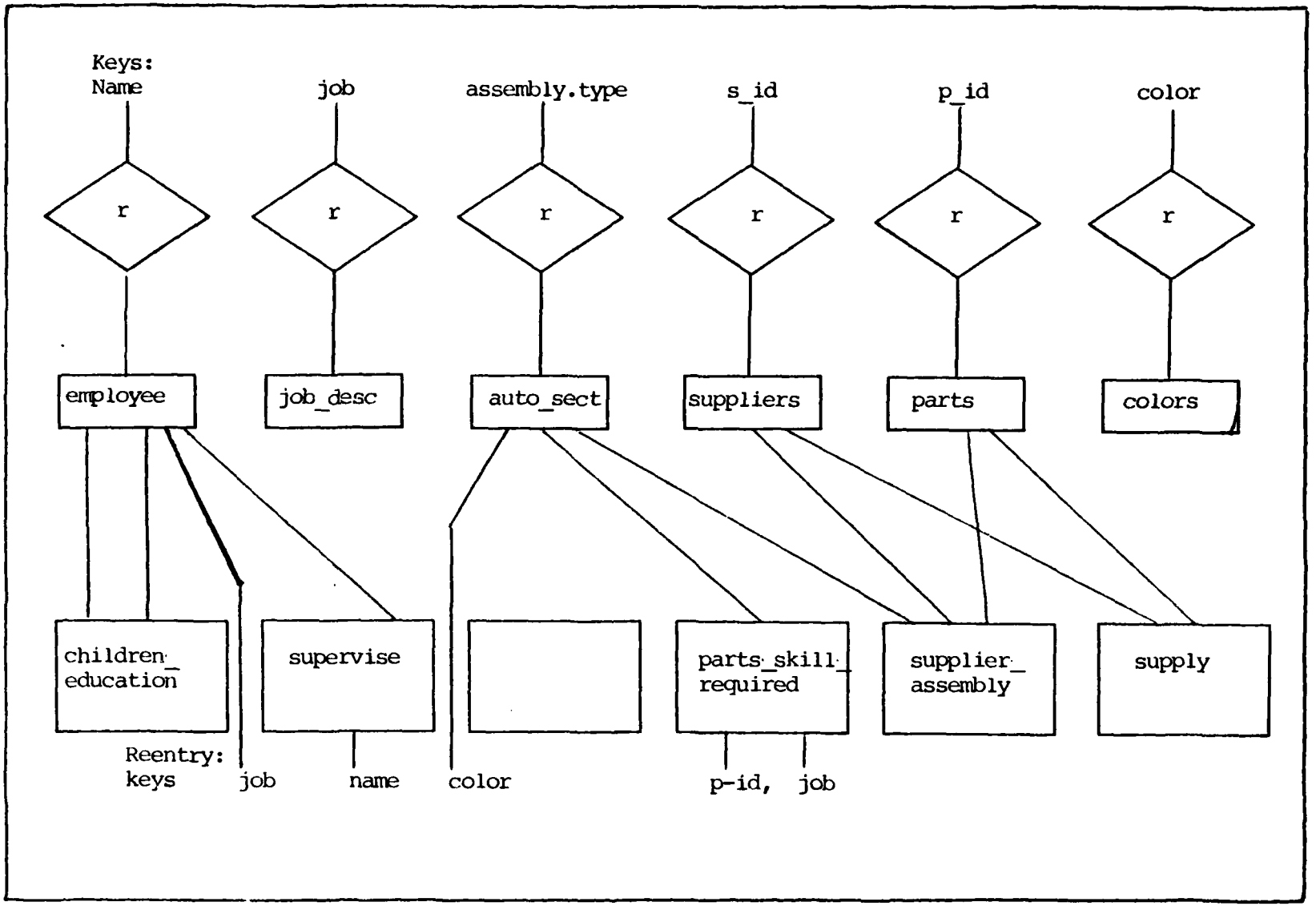


Fig. IV-1. TOTAL Network Structure

The top-level relations are implemented as direct files, and the bottom level relations as chains. The chains are like rings without a final link back to the top-level record. A record of a relation at the bottom level may be in multiple chains in order to form associations of entity relations, as shown by the supply record in Figure IV-1 (14:435).

Multiple chains may emanate from one entity tuple and can refer to different record subtypes within one subsidiary relation. This allows simulation of the required three-level hierarchy of employee.children. education by a two level hierarchy employee.children_education as shown in Figure IV-2.

Reference relations and lexicons are best placed in the role of entity relations at the top level. When they refer in turn to relations on the bottom level they can use linkages. Linkages to implement additional levels have to be handled by the application programs, using symbolic references. Output attributes which provide arguments for re-entry into the network from the top are indicated in Figure IV-1.

All linkages maintained by TOTAL are defined in the schema. They are created and modified automatically when a record is inserted. The linkages are not essential; a complete ruling part is kept in all bottom-level records. Since the chains do not return to the parent or owner records, the key in the ruling part provides a symbolic reference argument, which is the only means to locate the parent or owner of a bottom-level record.

Deletion of an entity record is not permitted as long as chain members are linked to it, so that in the example the relevant records in children_education have to be deleted before the employee record can be

part of structure. These structured references are refered to as links.

Links can be implemented by reference structures (pointers, symbolic, indirect) since they defer the binding and hence the existence of a network to query processing time (17:1151-1152). Direct pointer references can be used only if records are not moved within the database during their lifetime, since otherwise the pointers lose validity. Indirect pointers can be changed by changing the pointer index when records are moved. Indirect pointers are therefore the common means to implement linkages.

Pointer or indirect references may describe the structure redundantly because of the continued existence of symbolic references. If such pointer or indirect references are not redundant, however, this is sometimes refered to as essential links.

Loss or invalidation of a link implies loss of information. Maintenance files using essential links requires carefully worked out procedures to avoid validation of these links. In many network systems, links are essential although an application designer may decide to maintain redundancy of links keeping symbolic references within the data records (14:434).

A Simple Network Implementation

TOTAL provides two-level network hierarchy. The top level contains entity relations, and the bottom level contains either nest or associative relations. Figure IV-1 shows the placement of relations in such a structure.

Inability to obtain a piece of information from items already contained within the database (or the costly, lengthy search for an item) was also considered as a basis for inclusion.

Also, a data item determined from the personal interview process did not guarantee a place within the database if the item's marginal utility was not deemed sufficiently high. The AFIT Database was not established to duplicate individual's personnel records, although items like date of birth, place of birth and name of spouse were among those items included. For the same rationale, no student awards history data was expected to be kept other than data that related to those awards received while at AFIT, and then this data was expected to be used strictly for administrative accountability purposes.

Since a database has numerous requirements to meet, the selection and ordering of these atomic data elements for inclusion within the database was the next step in the design process. The entire project was developed using the TOTAL DBMS. Therefore, the database design conformed to the basic file structures and characteristics of TOTAL.

TOTAL DBMS

Initially, the primary questions that come to mind are: 'What is it?', and 'How does it work?'. A starting point is to say that it is a network database system.

A network is created when structures more complex than hierarchies are bound. A hierarchy is as complex a structure as can be built using ordering conventions for the segments. Even then reference pointers are found in many implementations. In a network references are an inherent

3. Failure to design for future data requirements.

4. Failure to design for future user expansion.

This allowed an important opportunity for the organization to obtain a new beginning. An additional consideration was to obtain the full support of the Data Base Administrator (DBA). This is of immense value when attempting a database reconstruction or update.

This technique contributed to unconstrained thought and far reaching visions for future database design concepts. The logical design process, almost be definition, is charged with taking a long-range view of the data needs of the enterprise (10:33). Appendix O contains a short explanation and proposed organization for the AFIT Database Administrator.

The use of the personal interview had numerous effects. The contacted individual felt as if their inputs aided in the design process. Because differing levels of supervision were contacted within the administration, a multi-dimensioned information and use requirement was insured. Also, no third party or intermediary was involved to taint the contents or results of the interview (no third party impressions of the interviewers).

After transcribing and refining all responses, the next task undertaken was to determine atomic data items retained in the database. The atomic data items were determined (Appendix C) based on the perceived user need for items of data (including supervisory requirements). The ultimate use of these selected values was not the primary selection criteria for their inclusion as an atomic data value. Items kept were determined by their value as a required piece of data, not just that they help satisfy a request for a utility or direct application program.

Chapter 4

IV. AFIT/ENG Database Requirements and Specifications

This chapter will review and explain in more detail the steps undertaken to effect the expansion of the AFIT/ENG database and will follow the transition of raw interview data through implemented database schema. The capabilities and shortcomings of the utilities used, TOTAL DBMS and the Form Management System, will be presented and explored. The steps utilized in interfacing these utilities with the required programming languages will be shown to satisfy the Software Development Life Cycle stages as well.

The initial step taken in this thesis effort is to determine exactly what information is required to accurately reflect the database design. A description of the database requirements is necessary to design an integrated database system. This means collecting the information used to generate the processing programs that deal with the database. As an example, if an operational database is available, using a conventional file system, we can begin by reviewing the usage of the file system by the programs that operate on the fragments of the database (14:369).

Personal interviews were conducted and the results tabulated (see Chapter 2). This technique eliminated a number of possible pitfalls.

1. First and foremost, not realizing the perceived need of the database.
2. Utilizing an old database design without realizing what changes may require data updates.

certain items of data were solicited during the design phase of the project.

Summary

This chapter covered the techniques utilized in the design of the AFIT database expansion. The method(s) used consists of looking at three levels or tiers:

1. Logical design
2. DBMS selection
3. Database Engineering

The stages of system development and system priorities were also presented for consideration in the selection and implementation in the design of a database.

4. System development time - how much time has been spent on background development
5. System availability - when data may be accessed, not necessarily when 'system' is functioning
6. System flexibility - important since future change or modification of any system is anticipated
7. Data integrity/security:
   - Integrity: Are there requirements for controls, journaling or audit trails?
   - Security: Is the data in the database sensitive? Is encryption necessary?
8. Reliability/availability of information:
   - Reliability: When data changes, how soon does user need to have updated information available?
   - Availability: Does the user have changing requirements to have data presented in a number of ways?

(15:65)

Although all the above items are important, not all may be as important at a given time. However, all these points were utilized as guidelines as much as possible during the design and implementation of the database. Since the project undertook the building of a database, it necessarily included the tasks of data collection, data organization, and data storage. Data organization and data storage were items mostly dealt with during the project. Data collection, because it is routinely handled by the administration, was not directly dealt with during the course of the project. As stated earlier, however, their needs for

the information model to a physical storage model), and determining program specifications. The database structure is also diagramed to provide a better picture, not only of the basic structure, but also to better project relationships among the data-sets and to act as a reference for the programming effort yet to be addressed.

The final two stages, programming and testing, by far take up the remainder of the effort. Inherent within this coding effort is the necessity to code for device media control language and the data definition language (DDL). Having a structured database is not the end, however. There still exists the requirement to code for data manipulation language and program logic, and load the program (16:98). The user only is aware of the tasks the database can perform, not the process which must be completed prior to the first application program executing.

With regard to cost, the resources of time, manpower, etc., spent on analysis and design prior to programming are inexpensive in the long run. For purely economical reasons, discovering poor design practices is best done as early in the development cycle as possible (16:15).

System Priorities

Priorities must be established for the development and operation of the database. Each priority must then be weighed as to their criticality for use within the database. The following items are among those which are important considerations:

1. Response Time
2. User function - ease of use
3. Volume of transactions

does nothing to change the effects or requirements for the other two parts. Each part must still be determined and subsequent conclusions drawn.

System Development Stages

In one form or another, most systems generally proceed in the development as shown in Figure III-1.

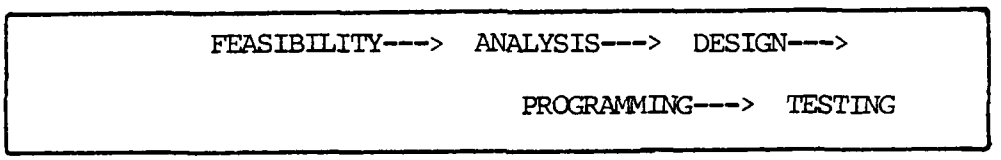


Figure III-1. Typical Stages of System Development

Feasibility is determined by 'management' (the administration in AFIT's case) to insure that the proposed system is both desired and feasible. This takes into account such items as the importance of the effort to the school, projected hardware and software requirements, the time and people resources required, as well as other factors. Once this receives a 'go' decision, the more detailed analysis stage begins.

The analysis stage involves determination and definition of the user's data and processing requirements. The user's information model (definition of data items and groups and function, definition of input to the system, output from the system, and how data is manipulated) is specified.

The design stage then transforms the information model and function into a format from which programs can be written. This process involves defining the data to the database management system (i.e., converting

the beginning or the end of a chain) or a relative location can be specified for retrieval.

All data entry, update, and deletion operations are performed by the CALL facility of the DBL in TOTAL. Records added to a single-entry file are mapped into a data space on the basis of their key values. Records added to a variable-entry file may be appended either to the beginning or the end of a chain of inserted within the chain.

Update facilities for both file types are similar. Each has a single write command for logical record access. TOTAL provides Macro calls to search single-entry files either by key or serially from a defined record position; by linkage-path chains, in the case of variable-entry files. In addition to these logical update facilities, a variable-entry file may be updated serially.

Records may be deleted from either type of files. In both cases, the deleted record's space becomes immediately available for reuse. However, all variable-entry records linked to the single-entry record being deleted must be individually deleted prior to deletion of the master record. This protects the records in the variable-entry file from being orphaned by the deletion of a master record.

Application Flexibility

TOTAL file organization is ideal in an application environment in which serial processing of small groups of records with common key values is required, since keys other than the master key in the single-entry record may be used as the basis for linkage paths in the variable-entry files, i.e., in AFITDB, if a single-entry key is SSN then the

set of all associated records in a variable-entry file has a particular SSN in common; hence are chained in a list. This feature is well suited to satisfying query conjunctions under certain sets of conditions.

File Maintenance

TOTAL performs several file maintenance activities in the AFITDB which are transparent to the user. These include chain counts, pointer control, and key creation or deletion. These controls ensure that the single-entry and variable-entry files retain their compatibility and that established chain linkages are not broken.

Whenever a record is added to a variable-entry file, several events occur. First, the new record is physically added to the file. Each master file linked to the detail file is checked to see if the key value already exists. If it does not, the key value is added to the single-entry file. The counter is set to 1 and both forward and backward pointers are initialized to be the address of the new record. If the key already exists on the master file, the counter is incremented. If the record is to be added at the chain's end, the backward pointer is used to access the current last record of the chain. Its forward pointer is set to be the address of the new record. The master record is also updated with the backward pointer holding the address of the new record (now the last one on the chain). The new record's backward pointer is set to the old last record. If the record is added to the middle of the chain, the master record's pointers remain unchanged. The new record reflects both forward and backward pointers. The record

preceeding and the record following the new records are changed to point either forward to or backward at the new record.

When records are deleted, these steps are reversed including decreasing the chain count and resetting either master or other chain members forward and backward pointers. If a single-entry file is used only as a key file (no data) and all of the counters are reduced to zero, that is there are no chains, then the master record is removed from the single-entry file. It is not possible to remove this record as long as the counters sum to greater than zero.

TOTAL Opportunities

Beyond the basic TOTAL characteristics, this package offers several opportunities for creative design. These include single-entry files with data, coded records, and multiple access paths.

Single-entry files always contain a unique key value. The records are located by a randomization of the key value. Data may be stored with the key if certain conditions exist:

1. Data has one-to-one correspondence with the key
2. Data is not desired by an alternative key
3. Data usually occurs for the key.

If data is stored with the key, the information is available to the program simply by locating the single-entry record by randomizing. In those cases where this approach is feasible, it will reduce the I/O necessary to obtain the non-key data.

Variable-entry files may contain one type of record or several. If several different record formats appear in one file, they are

distinguished by a type code. Therefore, these records are called coded records. Coded records are a fixed length as are all variable-entry records. If record formats differ in length, then some disk space will be lost. The advantages of coded records are:

1. Records of one key grouped together
2. Reduction in I/O
3. Ability to obtain one record by a coded read
4. Simplification of database design.

Records of different types could be placed in different variable-entry files, however, placement in one file allows all records for a given key to be grouped together. When these records occupy contiguous areas in a block, the I/O activity will be reduced. I/O activity is also reduced by eliminating the need to go to the single-entry file to pick up additional chains to follow. Records for one key which are grouped together are not always desired as a group. TOTAL has a technique called a coded read which allows the application to select only those records of a certain type.

The DBMS provides opportunities for multiple access paths into the variable-entry data records. Any field has the potential to be a key. This provides flexibility in programming batch applications and in on-line inquiry. The inquiries can use the key which will most quickly provide the information required. The batch programs can use alternate keys to reduce sorting. Secondary keys also provide access when the primary key is unknown. While every field may be a key, it is not desirable to have all fields as keys. If there is little probability that the data element will be used as an access path for batch or

on-line programs, then it should not be a key. There is an overhead cost associated with the addition of keys both in terms of disk storage cost and additional file maintenance. Every key requires a single-entry file to hold the key, its pointers and chain counter. Thus, this project determined keys using the relational Third Normal Form technique, to assist in insuring that such items as student social security number rather than course number were utilized to find a specific individual's education plan.

TOTAL Limitations

The DBMS does have limitations as itemized below. While the list of limitations appears lengthy, it should be remembered that a limitation is not a liability if the system being designed does not require that particular function. Limitations of TOTAL are:

1. Overhead in disk storage due to pointers, counters, keys and single-entry files.
2. Overhead in production time, expended to maintain pointers and single-entry files.
3. Inability to model deep hierarchies.
4. Requirement of fixed-length records.
5. Difficulty of generic, or stem, searches.
6. Disallowal of embedded or concatenated keys.

Overhead in TOTAL, as in any DBMS, is higher than for sequential or random files. Additional disk storage is required for the single-entry files which is a repetition of the key plus a pointer field and chain length counter for each chain into variable-entry files. Records

which repeat occurences on a variable-length record must repeat the key with each record occurence when mapped to a variable-entry file. The variable-entry files also must contain pointer fields for every chain which the record is on. Additional disk storage may also be required for coded records which vary in length.

TOTAL also has overhead in processing time due to file maintenance. Whenever records are added, deleted, or changed, pointers, counters, and key files are also affected. Retrieval of the affected records in addition to the actual change and rewrite costs additional time in processing. The overhead issue is not unique to TOTAL and is, in fact, a universal problem with DBMSs. The other drawbacks are characteristic of DBMS, which have a network approach.

TOTAL models parent-child relationships very nicely with the single-entry and variable-entry files. Deeper hierarchies than this are difficult to model. During the design phase of this project, no cases requiring more than a two-level hierarchy were determined.

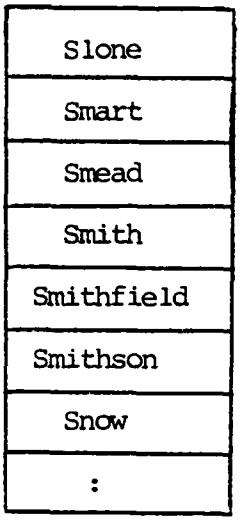
TOTAL requires that all records in both the single and variable-entry files be fixed-length. Stem searches are not possible with TOTAL directly. A stem search uses only a partial key and retrieves all key chains associated with the partial key. Due to the randomization of the key for placing the data in the single-entry file, it is highly unlikely that adjacent records are in sequential order. Sequential order in the key file is required to allow a generic search. If it is necessary, the problem does have a resolution. In addition to keeping the keys on a single-entry file, they should also be on a VSAM (or other indexed type) file. The only data on the file would be the key.

When a stem search was required, it would be performed on this indexed file. All records retrieved would then be used as a whole key in the database. This process takes longer and has some other drawbacks. The additional file requires file maintenance in the programs for updates, adds, and deletes. This is not automatic in the DBMS. The file also requires additional space. There is the added problem of maintaining compatibility between the indexed file and the database. This approach will solve the problem but should not be implemented unless absolutely necessary. Figure IV-3 illustrates this process.

TOTAL also does not allow embedded or concatenated keys. This means that the last four digits of the individuals SSN, if used as a key elsewhere, cannot be used as a key if SSN is also a key. This problem is solved by repeating the last four digits as a separate field if both paths are necessary. Concatenated keys would be a combination of two fields such as SSN and DOB. This problem also can be solved if need be by repetition of data in each variable-entry record (18:3-11).

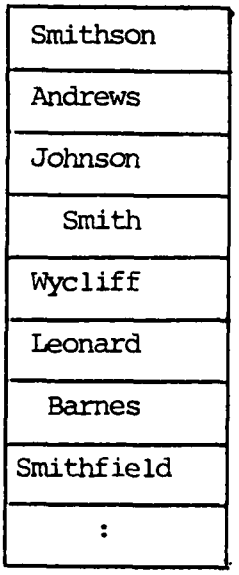
Data-Set Selection

The data items determined from the interviews were selected and grouped as either master or variable data. Master corresponded to items which tended to be less dependent on change (like course name, course number, SSN and student name) while variable tended to pertain to items less resistant to change and items determined by application process not requiring or unable to be kept permanently (like a student's selected courses which may change each quarter).



Indexed File

The stem SMITH results in selecting SMITH, SMITHFIELD, and SMITHSON from the indexed file.



Variable-entry File

Single-entry File

The whole name is used as a key into a single-entry file and the chain to its variable-entry file is followed as in the usual process.

Figure IV-3. Stem Search Under TOTAL

Appendix F lists the specific requirements from the directed interviews (on the left side), with the subsequent location for the inclusion of this data in the Data Definition Language expanded database generation directly opposite (on the right side). It is a refinement of those items, thought to be candidates for inclusion in the database, form interviews (Appendix D), and those items already existing in the database (Appendix E). The items in Appendix G were not necessarily requested through interviews, but added by the designer for anticipated use by the database. Items such as the 'Expected AFIT Departure Date' of a Faculty member and which courses belong to a particular course sequence were envisioned to greatly assist the Administration in future database requests. It must be pointed out that although the interview technique was conducted to gain important guidelines and insight, relying too heavily on it would create the same type pitfall as simply utilizing data items already contained in the database (see page 4-1).

Likewise, due to the synergistic result from comments of the Database Administrator (DBA), supervisors and users alike, continued changes to the design yet occurred. As an example, rather than have two separate variable data-set files for student and faculty Education History, a combined data-set was established (built). The same procedure was done for Awards. This action was taken to reduce the number of data-sets in the database, since the anticipated number of items versus their frequency of access was deemed low.

Anticipated Application Programs

Appendix D lists the major response areas determined by the interview process and depicts their priority for database development. Combining Appendix E, the original AFITDB data-set items, and Appendix F, the listing containing which new data-sets fulfill the proposed expanded database requirements produced the anticipated requirements for application programs as specified in Appendix H. Each requirements presentations item subheading is further divided into specific anticipated application programs to fulfill the listed requirements. These guidelines encourage better program database interaction promising substantially more comprehensive and viable database utilities for use by the Department of Electrical Engineering and AFIT.

Data Definition Language

After Stage One of the Design process is completed, the work of structuring the Data Definition Language takes place. The collection of information that describes the database, when organized in a formal manner, is called the schema. Data-element descriptions are an important part of the schema.

A set of documents that is used by a programming staff to generate programs could be viewed as a schema. Programmers will perform an analysis of the tasks and consider the available data elements. Subsequently they can code the required programs. Many statistical systems have provided directory facilities with their data files, so that the collected observations will always be properly identified and titled.

Figure IV-4 places the idea of a schema in perspective. The schema is used both to place incoming data properly into the files and to locate requested data at a later time. The dictionaries in the schema aid the users in describing their requests, and perform a filtering function to improve data quality within the database. The description of the database using the schema precedes the use of the database. Database users usually do not (nor should they) have the capability to modify the schema during database operations. In order to create a schema a language facility separate from that which is used when manipulating an existing database (14:377) is usually required.

In the TOTAL DBMS, the DDL outlines and defines space for the inclusion of data into the database. It also outlines the links among data sets and is the 'schematic' for the database as well. For this reason, a schematic was constructed to keep track, not only of individual data-sets, but also their numerous links, while relating atomic values with proposed application program requirements. Figure IV-5 depicts the AFIT Database prior to the revision and update accomplished by this project. The resultant schema (Figures IV-6(1) - IV-6(3)) depicts the expanded AFIT Database as well as the many data-sets and links to data-sets.

Database Descriptor Module

Utilizing this schema and the DBMS (TOTAL) DBMOD utility, DBGEN, a Database Descriptor Module (DBMOD) was generated. The DBGEN utility reads the Database Descriptor Language (DBDL) statements, and generates the source program. The DBMOD object file is provided after assembling

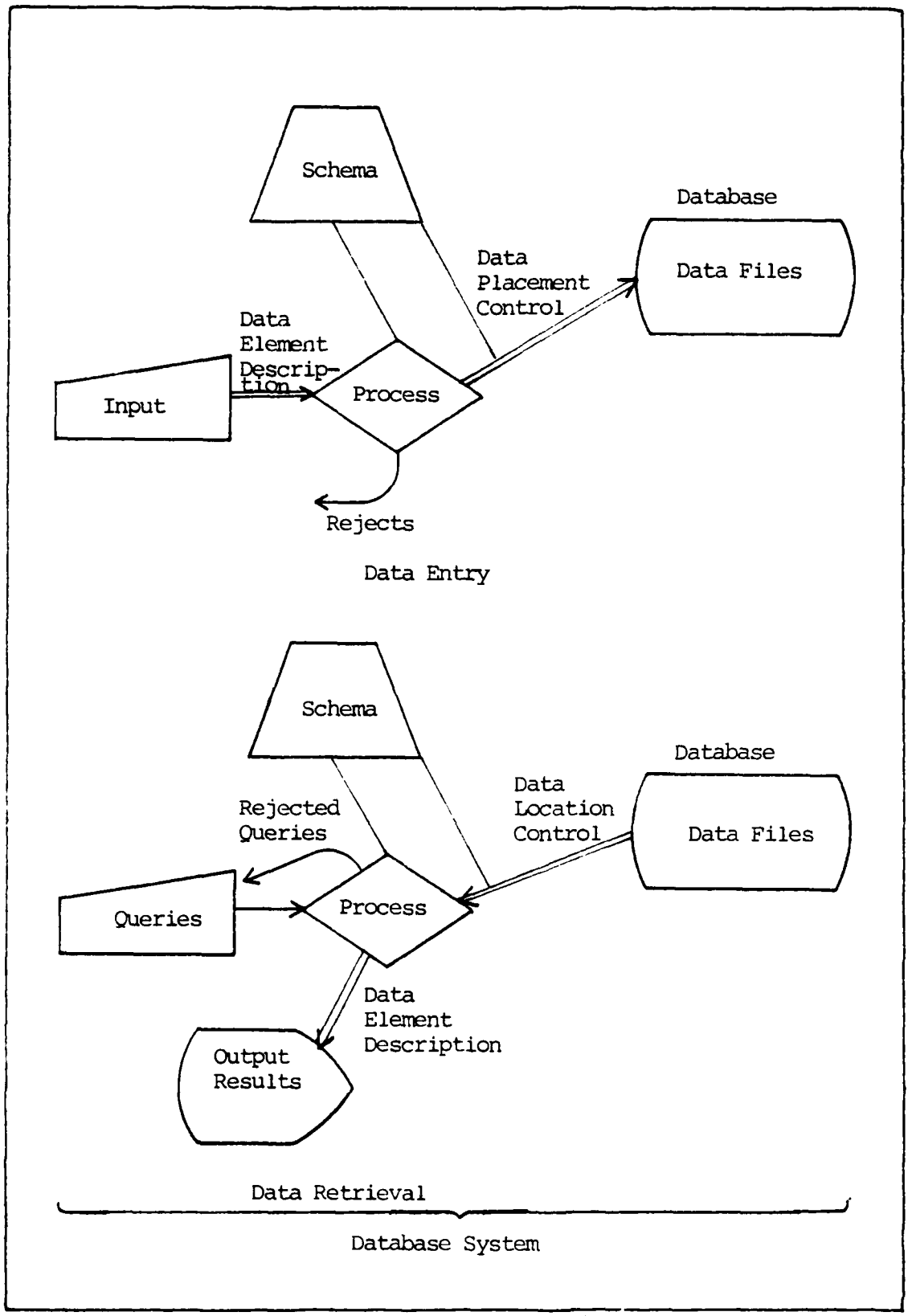


Figure IV-4. The place of a Schema in Input and Output

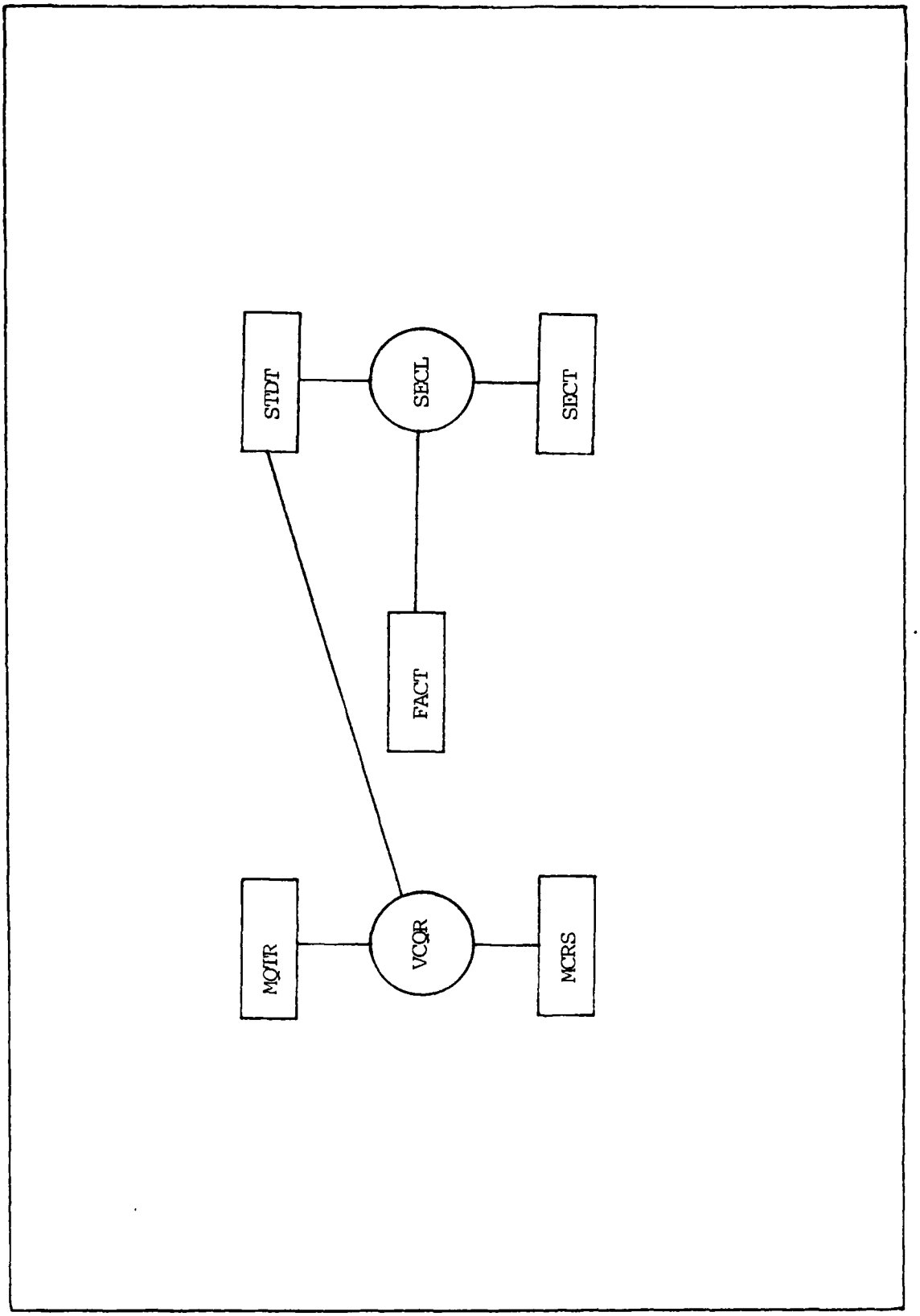


Figure IV-5. Old AFTDB

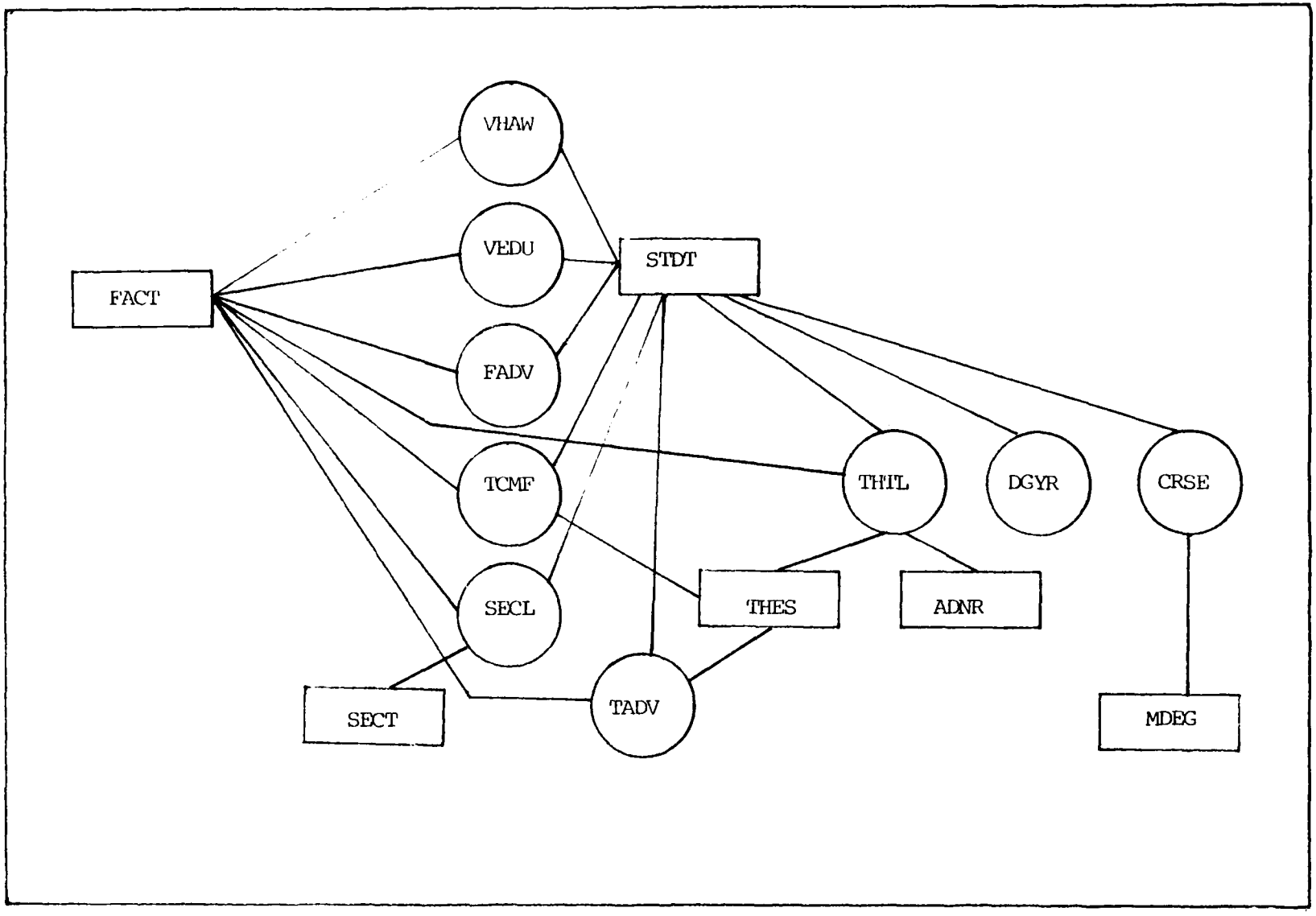


Figure IV-6(1). Expanded AFITDB

the area it represents and provides a further definition to the user of the screens from which he is expected to choose. The user then selects the next data-input screen corresponding to the specific data required for input or viewing. The approximate screen corresponding with his selection is then displayed and the terminal remains open collecting input. The method utilized to move around within an FMS screen is by use of the 'Tab' and 'Backspace' keys. The 'Tab' advances the cursor from field to field while the 'Backspace' key backs up the cursor from field to field. Other keys, like 'Delete' (which removes characters from the display) and the arrow keys (which move the cursor in the direction of the arrow within a field) are obvious, while the 'Linefeed' key is not (it blanks the entire field). Further specifics can be found in the VAX-11 Software Reference Manual (19).

The Driver was developed with not only professors/instructors or students envisioned as users, but also the secretary charged with inputing all the initial data and subsequent updates. This can be a tedious undertaking for the secretary. Therefore, after the last screen for the chosen area has been exited (some areas call for more than one screen to be displayed in succession), a final screen is displayed which prompts the user to choose among three choices.

1. Input additional data utilizing the last series of screen(s)
2. Return to the previous (last) menu displayed
3. Exit the program (Figure V-4).

The choice of entering additional data closes the previous data input file prior to opening a new file for the additional input. Program 'Driver' does have the option to exit at the 'FIRST1' screen, but

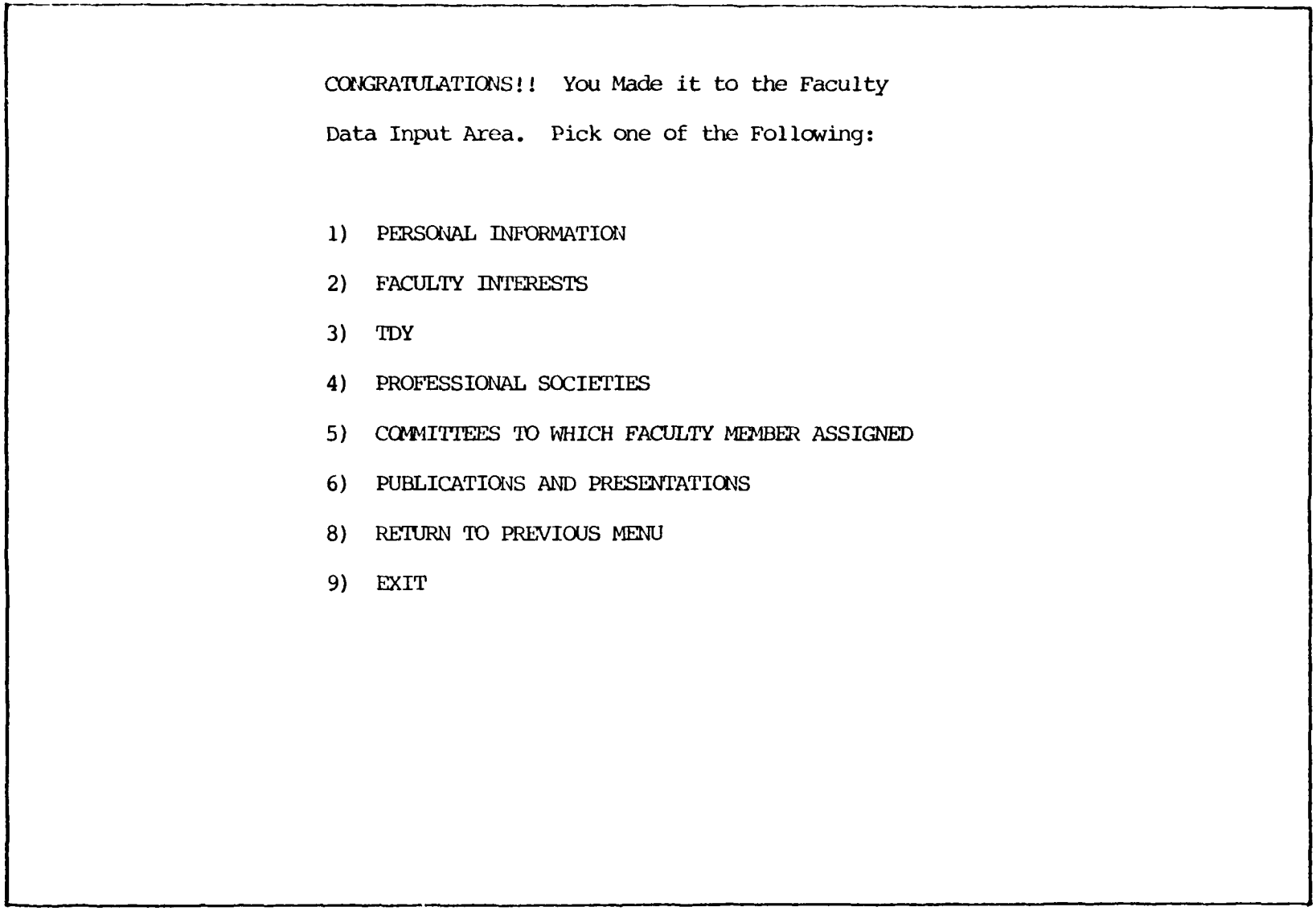


Figure V-3. Faculty Data FMS Screen

Select One Digit to Choose Which Area You Desire
to Enter Data Into AFITDB.

1) FACULTY (Personal Information, Intersts, TDY, Professional Societies, Committees, Publications and Presentations)

2) STUDENT (Personal Information, Courses in Ed Plan, Thesis Information)

3) STUDENT/FACULTY (Education History, Honors/Awards)

4) TEXTBOOK INFORMATION

5) SCHEDULING INFORMATION

6) OTHER MASTER ENTRY DATA

7) UTILITIES (Ed Plan, Degree Requirements, Student Graduate Credit Record-AFIT Form 89)

9) EXIT

Figure V-2. Menu Screen 'FIRST1"

AFITDB DEMONSTRATION MENU

Enter one of the digits below to access the listed function:

1) Input Data Only

2) AFITDB Database Utilities

9) Exit

Figure V-1. Initial Menu Screen

Nowhere in the reference material is this most important fact listed.

Form Driver Library

The Form Library, AFITLIB2.FLB (Appendix I), was designed to hold menu and data display/collection screens patterned after the Schema for the expanded database. The first screen is 'AFIT1' (Figure V-1) which is an introduction screen and asks for a choice to input data, use utilities, or exit. Since the screen was designed with an expanded database in mind, the first two options call the same next screen, 'BEGIN'. This screen can be used to check user security, if desired, or to prohibit the user from advancing into the database any further.

The next screen, 'FIRST1', is the first real user menu. Here the user may make one of seven choices:

1. Faculty (Personal information, interests, TDY, Professional Societies, Committee's, Publications and Presentations).
2. Student (Personal Information, Courses in Education Plan, Thesis Information).
3. Student/Faculty (Education History or Honors/Awards).
4. Textbook Information.
5. Scheduling Information.
6. Other Master Entry Data.
7. Utilities (Ed Plan, Degree Requirements, Student Graduate Credit Record Form (Form 89)) (Figure V-2).

A choice of any of these screens will cause that menu screen for the chosen area to appear (Figure V-3). Each menu screen announces

a major time consuming effort because of the unavailable PASCAL FMS documentation and VAX/VMS FMS version update required to build and run a self-contained and directly FMS-related structure.

4. Debug Program with Forms - This followed the necessary work to incorporate the ability to capture data input and retrieve data through the DBMS and the operating system. Existing application programs were modified to accept this utility, which required compatibility debugging.

Initially, a number of forms were created for use with the expanded AFIT database and placed within a library, AFITLIB2.FLB (Appendix I), for use with the 'Driver' routine to test both the form capability and the unique call requirements. This test-bed was invaluable for determining inconsistencies which resulted by using the reference material and provided examples. For example, the reference states that for the specific call 'FDV$NDATA', the call "...is used to access by name data that has previously been associated with a form as named data" (20:6-13). An example of this call as used in an example program is (the FORTRAN statement): 'IF (FDV$NDATA (RESP, FORM) .GT. 1) GO TO 90' (20:A-16). It was not until later that it was discovered that this call does not return a value unless specified by yet another call and that value is FMS specific (which is related to a predefined error coding which can be a positive or negative number) (20:5-3). To effectively use this call, the value returned must be used as a Boolean, and cannot be used to compare values.

Chapter 5

V. AFIT/ENG Database Design/Implementation

This chapter will highlight the remaining stages within the Software Development Life Cycle as pertain to this project. The integration of the database design with that of the coding and unique display technique will be explained in detail. Additional stages in the Software Development Life Cycle, from coding through testing, integration, and acceptance testing, will be highlighted to show the transition within the effort.

Utlizing FMS

A short development cycle for the inclusion of this utility with the database had four stages:

1. Plan - This was usually short, since the initial database design fairly determined the format for the form. Determining the atomic values tended to outline this most accurately.
2. Design Forms - This stage followed directly after the Planning stage, and was determined quickly due to the planning stage rationale.
3. Write and Document Program - The initial work done was to develop a trial Driver to insure understanding of the specific commands and their relationship with the programming language. Additionally, compatibility experimentation was required to be conducted to also determine the means required to interface with the DBMS. This became

(item 5) and service (item 7), precede the student's place of birth (item 22) and his previous command (item 28), which will not conceivably change while at AFIT. More specific data on FMS is contained in Appendix O.

Summary

This chapter characterized the TOTAL network DBMS and its capabilities and limitations, the Data Definition Language (DDL), the design settled upon for the schema and data-sets, and provided an overview of the Form Management System (FMS). The time spent on the many design revisions and structuring required for the DDL and schema pay dividends later in the project. Although minor revisions continued to occur as additional data was included in the project, these were minor because of the in-depth analysis already completed. As in the Software Development Life Cycle, as integration of the project parts occurs, the worth of the design stage is borne out, for the errors introduced in the design stage are detected in the integration stage.

Student History

Enter the Following Items:

1. Student Social Security Number (9): - -
2. Graduated AFIT (Y/N):
3. Student Name (28):
4. Box Number (4):
5. Mil/Civ Rank (3):
6. Education Code (5):
7. Military Service (2):
8. Date of Rank (6):
9. Duty AFSC (6):
10. Primary AFSC (6):
11. Date of Commissioning (6):
12. Years of Service (2):
13. Sex (1):
14. Race (2):
15. Religion (20):
16. Duty Phone (7): -
17. Aero Rating (10):
18. Current Address (40):
19. Emergency Address (40):
20. Home Phone Number (7): -
21. Date of Birth (6):
22. Place of Birth (40):
23. Marital Status (1):
24. Spouse First Name (12):
25. Spouse Date of Birth (6):
26. Military Spouse (Y/N):N
27. Number of Dependents (2):
28. Previous Command (5):
29. Duration (2):
30. Last Organization (50):
31. Last Position Title (50):
32. Do You Require Additional Entries of This Form (Y/N):

Figure IV-7. Student Personal History Data

Therefore, menus were selected as the most user friendly mode of operation for both data input and retrieval.

An available utility on the existing hardware is the software package from Digital Equipment Corporation (DEC), the Form Management System (FMS). It is designed for ease of formulation and form simulation in order to collect the transmit data in an orderly manner. Forms are designed by typing them directly onto the terminal screen. Neither layout charts nor a special forms design language is required. FMS associates constant data with the form, not with the application program, resulting in simplified application program maintenance and increased application program flexibility. Forms may later be modified without the need to recompile the application program (as long as the form does not change application program specific areas). This utility supports any language supported by the VAX/VMS host computer.

The order of the data contained on the FMS screens is determined by grouping data items by category, and, to a lesser extent, by screen requirements. Each screen was created to present cohesive areas for data input and to make judicious use of limited space. Screen requirements pertain to the data collection precedence with the FMS (top-to-bottom and left-to-right). In an effort to show all personal data on the CRT screen at once without having to scroll through an area to see it all, the selected option was to forego the use of scroll areas. All personal data is still grouped by category. For example, see Figure IV-7. The most often accessed student data precedes the data less often changed to accommodate the anticipated changes required to be performed by the user (secretary, student, etc.). Thus, a student's rank

space allocated for user use would be less than calculated. He also provided a means to determine the size of the schema which can be found as an octal number at the end of the .LST listing after the schema is compiled. Converting the octal number to decimal and dividing by two gives the size in words which can then be compared to the maximum permissible size minus the given TOTAL restraints, above. For this project, the additional work-space areas were deleted in order to validate the schema and test the FMS and application program integration. Subsequent revisions on the size of the database verified that at maximum size, only a few additional work-sapce areas would need to be deleted (see Chapter 6, Recommendations).

Form Management System

Along with the database design, and the use of the DBMS to act on the data, application programs to manipulate the database are required. With this goes the task of interfacing such applications programs (and database) with the user. Inasmuch as the design of this database and its use is for the present as well as the future, it was felt that utilizing an enhanced data presentation utility would further the usefulness of this database.

The main thrust of this database is storing and retrieving data items involving individual student and faculty personal history as well as student education planning. The interviews pointed out the large volumes of data required to be input with the start of each new class, subsequent course scheduling, as well as requested utilities designed to make use of the expanded database to a degree previously unable.

this source program. The DBMOD is now available for use with TOTAL and its utilities (6:3-4).

The size of the DBMOD was determined, initially, by the design of the schema. This design included additional workspace areas to permit the multi-user operation of the TOTAL database. However, each time the DBMOD utility was utilized with this schema, an error code corresponding to a size limitation was displayed. A literature search did not alter this continued error display, and a telephone call to CINCOM, the TOTAL developers, finally obtained a solution.

Although the TOTAL installation documentation (6:2-17) states that the maximum extmod size is 32768 words, it does not state how users may find the size of their proposed database schema. The size required by TOTAL is 10,800 words with any logging options and communication buffers each requiring a specified amount. This total subtracted from the maximum, based on the documentation, should show the maximum user schema size. Each time the size error occurred, the TOTAL installation procedure was re-initiated in order to enlarge the extmod space. The additional size requirement was added to the last extmod size to arrive at the needed amount. However, entries of values greater than the default value specified (later determined to be the maximum accepted), generated the default value used as the extmod size.

The CINCOM software representative (17) explained that TOTAL can only handle a size limitation as outlined in the installation guide and even the newest version, yet to be released, would only expand this size limitation very little. However, he also stated that since the AFIT version of TOTAL included software patches, that the available

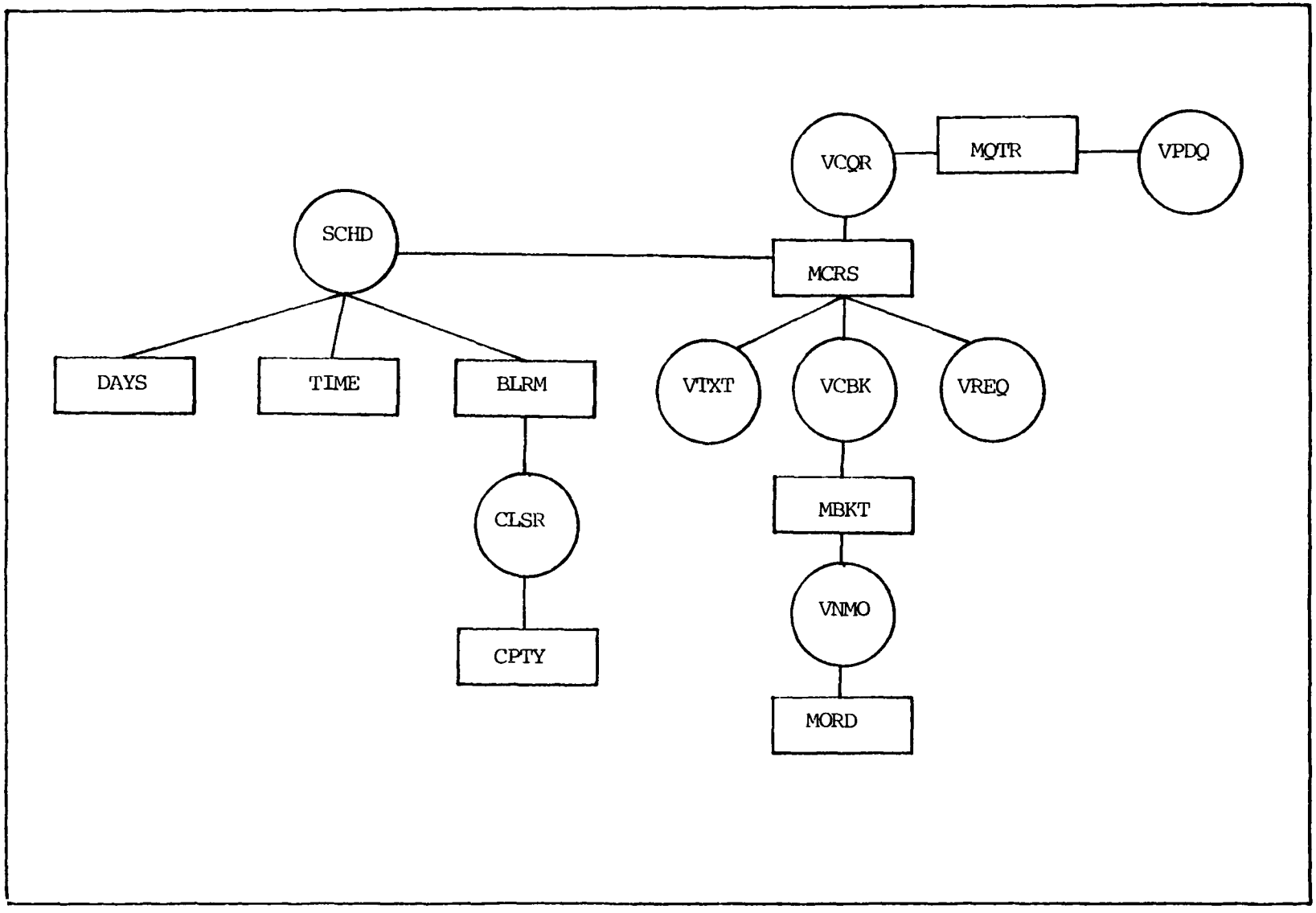


Figure IV-6(3). Expanded AFITDB

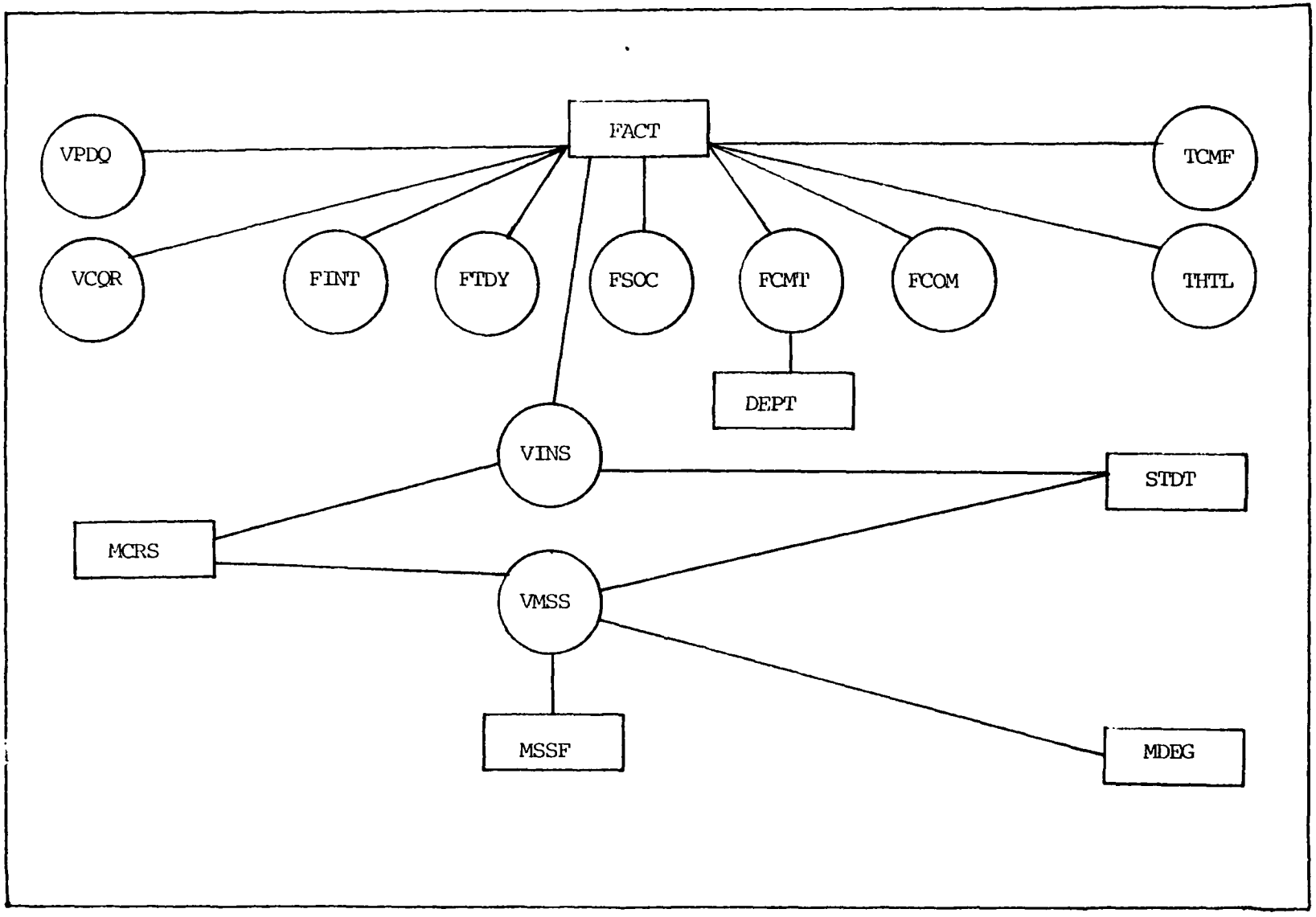


Figure IV-6(2). Expanded AFITDB

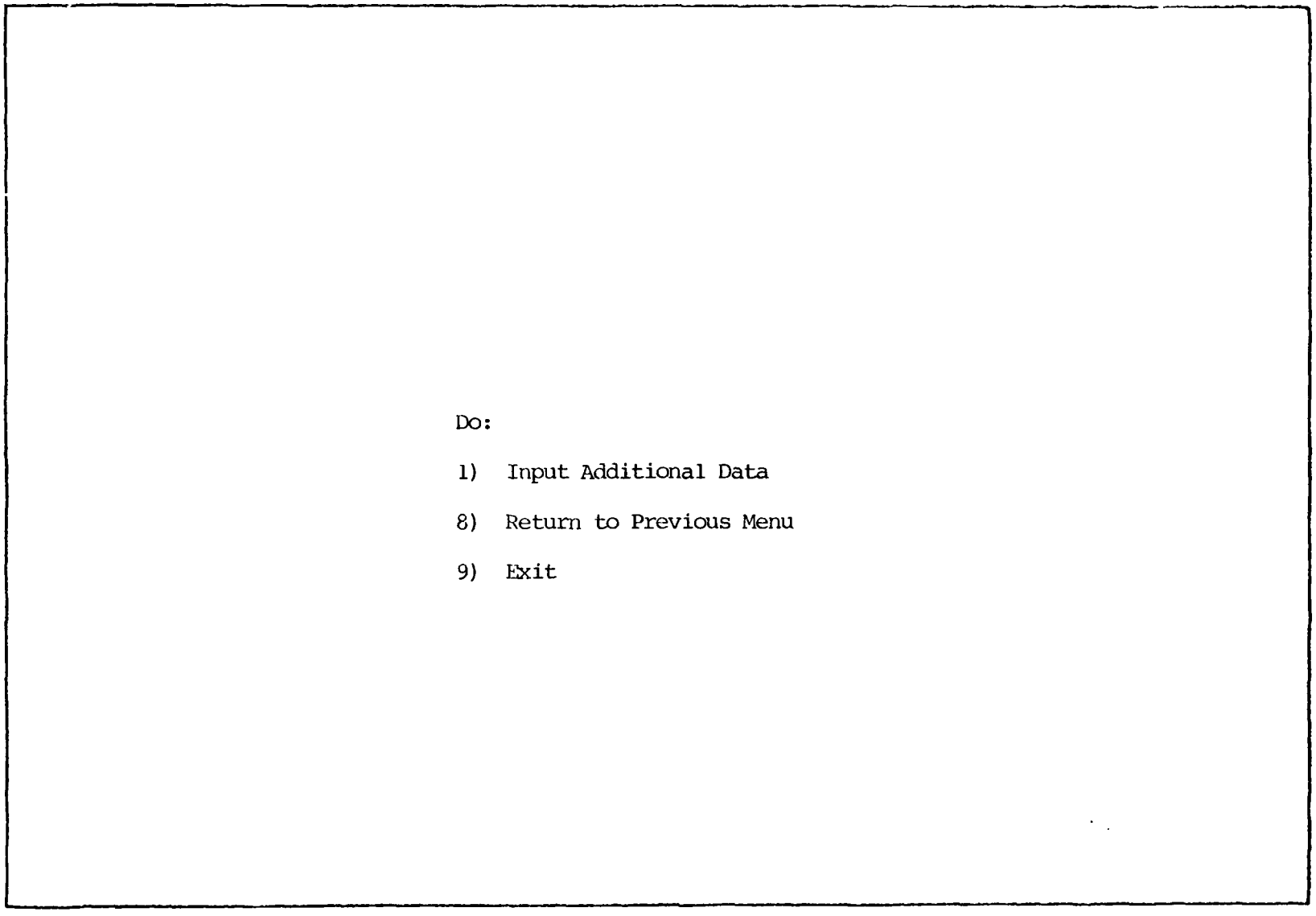


Figure V-4. Last Menu Screen

does not allow the user to back up past this screen, since screens prior to this are for entering the database only.

When 'EXIT' is the elected option by the user, a number of things transparent to the user happen. Since this option resides on a number of screens, exiting during a reverse video displayed screen would cause the users terminal to remain reverse video. Through a short routine, when exiting normally, the screen automatically defaults to black background, the screen is cleared, and the cursor is placed in the home position.

Appendix I lists the screens contained within the library. Not shown in Appendix I are continuation screens for Graduate Credit Record Form (Figure I-35) and Student Education Plan Form (Figure I-32).

Driver Program

It was essential that prior to jumping into designing application programs to pass data between the database and the user, that the ability to determine how this must be accomplished would first be determined. This, the FED and FUT were used in conjunction with producing a program able to call each form in the library and pass data.

The first step was to achieve a working Driver program to display forms and input data. Another problem arose at this point. Because no documentation was available to provide assistance as to the proper handling of parameters in PASCAL, the Driver had to be written in a language supported by some available documentation. FORTRAN was chosen because of its similarity to PASCAL. By utilizing the provided model (20), and example FORTRAN program (20:A-15), the 'Driver' program was written

(in FORTRAN) with its purpose to display each form in the library (see page 5-2). After determining how each Form Driver call both passed and received data, the Driver program was able to function as designed. Problems in working through program discrepancies aided in establishing what parameters would be required in adapting this test program to a working database application program (see page 5-3). Such coding and testing preceded the integration of the utilities with the operation of the database, which complied with following the software life cycle development.

The next step was to achieve the display of 'old' data (previously written to a file) from a file to a displayed form utilizing more Form Driver calls. This became a replay of the techniques learned from the first step (above), and was necessitated by the lack of documentation and errors in existing examples. This step was the logical extention of the first to determine the techniques required to infuse the FMS operation with the DBMS. This progam was called 'Input'. The major problems centered around retrieving data previously written to an external file, retrieving the data within the file and assigning the fields within this file for display through the proper screen. This test program, also done in FORTRAN, proved that the VMS calls properly functioned as intended.

The third step became a series of proofs for eventual Form Utility inclusion with the DBMS. The new database structure schema had already been designed and the Database Descriptor Module (DBMOD) generated utilizing the TOTAL DBMS DBGEN utility (see Chapter 4). The order of the

data-sets within the database generation schema show the master data-sets followed by the variable data-sets. This was necessary because any references to the master data-sets were required to first insure that the master data-set be known to the DBGEN utility. Thus, all master data-sets precede any references to the variables. There is not the same requirement that variables precede any other variables that they may reference.

At this point the DBMOD was initialized and data loaded into it in order to insure that this most important portion of the project functioned. Application programs which functioned with the currently used database were run utilizing this new, upgraded database version. Data was successfully stored and the application programs adequately run with them. This effort was temporarily suspended to determine whether the expanded AFIT database, previously created and compiled, would function as designed with the redesigned and rewritten application programs. The emphasis was placed on proving that the database would accept, retain, and output data.

Three application programs (NSTUDENT.PAS, COURSE.PAS, and INPUTFACT.PAS), which functioned properly utilizing the original AFIT database, were slightly modified in order to test their ability to obtain identical results using the the expanded database. The application programs were modified to display the FMS form screens previously created to determine what modifications would have to be made to incorporate the form utility within the existing application programs and TOTAL Database. Using the modified programs and the expanded AFIT database, the tests

proved that the new database performed as designed, at least for the test programs.

Database Design Steps

This series of steps was crucial in determining the proper scheme to interface the form utility with the DBMS and application programs. The lack of proper documentation and debugged examples, from Digital Equipment Corporation to use as guides, required additional design and programming time to gain the necessary knowledge in order to utilize this most useful and unique utility. Once installed, the ability to change or add additional forms, without adding huge amounts of accompanying code, became a much simpler exercise. In a matter of fact, utilizing the form manager frees the programmer from worrying how to both capture data for the database and write the application program within the same program. Once the application program is written, the form can be constructed; or, the program can be written after the form is completed. The additional coding required to place data into or retrieve data from the database, utilizing a displayed form to the (CRT) screen, can be included after the application program is structured.

FMS Version 1.0 (which was the only version available for use during the project) did not support direct calls from PASCAL, nor the ability to call from FORTRAN to PASCAL. This necessitated initiating the database session in PASCAL and calling the Driver program as a subroutine. Also as a consequence, data variables needed by both PASCAL and FORTRAN subroutines were passed back and forth in order to efficiently utilize the interactive ability to communicate directly from PASCAL to TOTAL (although this tends to violate good software

programming by having global variables, however few, running around). This technique was perfected after determining that a change in the FORTRAN data capturing 'WRITE' and 'OPEN' statements was necessary to keep from losing data during the passing of buffers written in FORTRAN and read in PASCAL with subsequent FORTRAN 'writes' on the 'read' file (this is documented in programs 'TEST.PAS' and 'TEST1.FOR'). Succintly, data captured into a buffer (utilizing FMS screens) was initially written to a file utilizing a FORTRAN write, which tended to drop the first character in each line since it was used as a carriage return read. Eliminating this quirk fathered numerous test programs. The method of writing to the data file was changed by inserting the keyword "Carriagecontrol = 'none'" within the 'OPEN' command (see program TEST1.FOR) to preclude this from occuring.

This series of steps became the crucial exercise in determining the manner in which the different media would interface. The DBMS contains the data, the FMS contains the forms through which the data is entered, and the application programs are the means to join the screens with the DBMS.

Enhancing the user's ability to quickly check data as well as his desire to manipulate it became the next important factor. The ability to utilize the FMS in conjunction with application programs became an integral part of the project. This thrust became a challenge to adequately package the FMS into the database utilities to support the expanded database providing a much more efficient and user friendly environment and do so in order to simplify the required application programming code. This work toward future expanded database usage

uncovered a myriad of problems requiring solutions in order to effectively utilize this package with TOTAL. On the other hand, including FMS reduced the amount of new and old code required to perform identical applications. Such enhancement thus allowed writing more simple application programs which utilized the more transportable FMS form modules.

At this point the technique required to integrate the PASCAL application program with the FMS written in FORTRAN (which would insure data transmission between modules) had been accomplished.

Following this, the next test consisted of merging the two previous results in order to determine the proper interaction and functioning of the different programs:

1. The PASCAL application program
2. The FORTRAN FMS Driver
3. The TOTAL database

NSTUDENT.PAS was selected from the initial test programs for modification and testing. Since the previous testing had established how to accomplish data transfer from PASCAL to the FORTRAN Driver and back, the program was modified to further incorporate interaction with the TOTAL DBMS and the expanded AFIT database. Throughout this test only small changes were made to insure continued progress without losing extensive amounts of time due to debugging caused by numerous possible coding errors.

An additional problem was discovered during the assignment of variables from data retrieved out of the database. Utilizing the FMS 'PUT' statement, data is placed on the screen through the form templates.

Even though particular fields are designated to display data only (only the application program may enter data in these fields), it is incumbent upon the programmer to pay particular attention to the placement of data since during this operation data may be displayed into any field, even those fields into which the user may not input data.

The resultant program, NEWSTDT.PAS, called the FORTRAN Driver, TESTNEW.FOR. These programs were modifications of the previous test programs (see page 5-13) with TOTAL interfaced for the first time. The many small changes required to achieve the ability to input student data into the database master STDT data-set and then to retrieve and modify the data quickly proved the test plan was valid. This test was completed in less than the week that was planned. The resultant reduced amount of code required to achieve the test results, as well as the concomitant ability to rapidly update master data-sets, was a direct consequence of using the FMS in conjunction with TOTAL DBMS.

Continuing to build onto the previous results, the NEWSTDT.PAS Driver was modified to begin the interactive session with the beginning screens within the library, and to determine the social security number (key) of a generic individual. Modification in this manner utilized existing code while expanding the capabilities of the fledgling expanded database utilities. This capability would allow either student or faculty data areas to be directly accessed by later application programs to better serve the needs of the frequent database user.

Consequently, the Driver, now designated NPERSON.PAS, returned the user request regarding either student or faculty data. In actuality, what was returned to the calling module was the user choice of the

activities offered on the 'FIRST1' form (see page 5-3). By allowing the module to return this data from the FORTRAN FMS program, redefined TEST2.FOR, the Driver was able to use the data to determine which modules to route the in-use user defined program. This use of a variable already used within the FORTRAN FMS program for use within a case statement in the new Driver program did not increase the number of variables within the program, although it did require another global variable. The trade-offs in non-specificity and clutter were acknowledged and accepted by the increased ability of the utility.

The use of the returned variable 'RESP' was subsequently used within the module 'Selectoperation' to route the program to separate FMS and PASCAL utility programs to accomplish each specific task and return to the calling module when completed. Although limited in operation, the addition of application programs, in the form of modules, promises to be a more efficient and user friendly operation (using the FMS utility) than requiring the user to know about and call each separate application program, requiring built-in code for input or output of data, which was the manner of operation which existed prior to the beginning of this project. The PASCAL Driver program is designed for modularity and expansion, necessitating the programmer code only utility specific application programs and little additional coding required to connect to the main program Driver.

Test Plan

A formal test plan to test the validity of the system design for the TOTAL DBMS Driver program has been developed. A set of specific

tests for the generalized design was developed using the functional requirements found in Appendix M. A sample test module is shown in Figure V-5. The Test plan contains the functional requirement, special test cases, expected results of the test cases, actual results of each test case and a remark section. The complete test plan will be determined based upon the application programs and the DBTA (see Appendix N). Throughout the coding and implementation of the Driver program, test procedures were conducted using the individuals who had been interview subjects at the beginning of this project. When warranted, additional changes to design were made after such discussions (such as allowing the user to exit the Driver program prior to the 'FIRST1' screen, before the 'Signon' procedure).

Test Procedures

The test plan presented was implemented using incremental testing based on software engineering (24). The test procedures used unit, integration, validation, and system testing (19:295-305). Unit testing ensures each module performs as specified. Integration testing ensures modules of a program communicate correctly with each other. Validation testing ensures the program meets the functional requirements. System testing ensures all software in the system performs correctly and meets the functional requirements.

Summary

At this point the main thrust of the project was achieved. By utilizing the techniques for relating the different means of achieving this blend of powerful DBMS, extremely user and programmer friendly

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 1.2 - Allow user to select type of database operation.

TEST CASE(S):

1. No selection entered.
2. Wrong type data entered.
3. Exit selected.

EXPECTED RESPONSE:

1. Unless exit selected, user unable to advance until proper response made.
2. Data not acdepted for input, terminal signals this to user in form of bell.
3. If choice is exit, shut down database and log off.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____
CASE 3. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS

(23:APPENDIX D)

Figure V-5. Sample Test Plan

FMS utility, and application programs, a very powerful and more easily maintainable system was created.

This chapter characterized the TOTAL DBMS, provided an overview of the FMS utility, and presented the steps utilized in interfacing the above with the database schema, DBMS, and programming utilities. Some of the problems and solutions to those problems are mentioned in an effort to assist future work on the database. The coding and testing described in this chapter resulted directly from the design determined utilizing the Software Development Life Cycle described in Chapter 1.

Chapter 6

VI. Conclusions and Recommendations

Introduction

This chapter presents the conclusions and recommendations derived from the results achieved by this study. In so doing, the Software Development Life Cycle continues to be referenced to highlight the efforts taken to develop lasting results.

Conclusions

The initial part of the project was to enlarge the AFIT database to include Faculty data portions. This was accomplished utilizing various personal interviews and practicing a technique to design an entire database as if one did not already exist. The time spent validating expected user's requests for specific data items and requested application program data items generated atomic values used to determine the expanded database configuration.

During the development of the Expanded AFIT database, the expected result was that this would be a multi-user environment. This underlies the Security/Integrity concerns which must continue to be shaped and monitored. Accordingly, the database was designed with these considerations in mind.

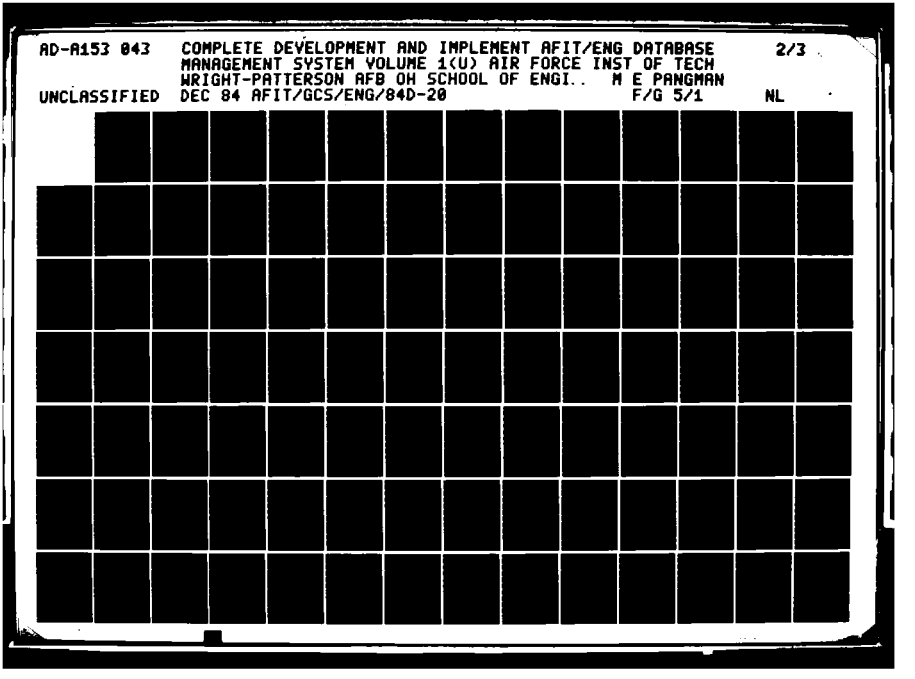
Multiple I/O areas were defined in the original design Data-Base-Generation for the purpose of making available additional multi-user memory work space areas. Multiple taskings of data-sets are not allowed under TOTAL unless separate work areas (scratch-pad-like) containing

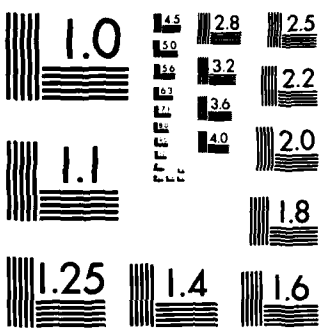
additional copies of the data-sets are available. Accordingly, the arrangement of the data-sets into groupings which would assist this multiple-user environment was acknowledged and implemented. Those data-sets believed to be most helpful within the anticipated user environment had additional work-space-areas designated to allow distinct user actions.

Working through the Requirements Analysis and Specification phase, the database schema was designed to accommodate the desired and expected taskings of the user. Testing the code defined by the data definition language required that it be able to store the input data and return data via links determined within application programs acting through this DDL.

The resultant database now contains the required data elements desired and requested to perform the myriad of additional tasks anticipated for it to accommodate. The use of the Form Management System as an enhanced data display utility answers the need for data presentation in user understood and easily repeatable format. In a conscious effort to assist the user in entering and checking data, scroll areas were not used. Thus, when entering many items of data onto a predefined, database provided, FMS produced form, the entire form may be easily seen and checked on one screen utilizing no scrolling to see all the data contained on the form prior to entering the data into the database.

The use of the menu format to page through the available database utilities is considered as the best means of user-friendly prompting and 'directing traffic' without excessive verbiage. The current menu listings encompass student and faculty personal data, book and


AD-A153 043 COMPLETE DEVELOPMENT AND IMPLEMENT AFIT/ENG DATABASE MANAGEMENT SYSTEM VOLUME 1(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. M E PANGMAN
2/3
UNCLASSIFIED DEC 84 AFIT/GCS/ENG/84D-20 F/G 5/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

scheduling information and a separate section for utilities (Student Graduate Credit Record, Ed Plan, etc.). The menu utility also permits the user a built in ability to back up, within the menu itself, if desired in planned application programs.

The most important features implemented for use by this expanded database is the inherent ability for easy expansion and an equally easy means allowing the user to find quickly the type of operation he desires to perform. Since the Driver program allows the user to determine the type of utility he desires to select before implementing any database usage, future database utilities can be written and added with a minimum of Driver changes.

The menu system set-up within the called form subroutine and utilized throughout the database utilities encourages the ability to build in utility and user specific security for the access of specific utilities or data items within the database. As the number of users and the type of data utility application programs expand, the security concern will no doubt also increase. Thus, the database and its surrounding form utility system will be able to keep pace. Some of the problems and solutions to those problems encountered during the course of this project were mentioned in the course of the documentation in an effort to assist future work on the database or its utilities.

Recommendations

The Software Development Life Cycle approach followed for the development of the AFIT Database includes provisions for additional coded, tested, and integrated application programs. Future work to

complete the application programs proposed during this project and to make use of the newly included atomic data items added for just such new programs (see Appendix H) should be undertaken to obtain the expected utility from this project. Work yet remains to complete conversion of the original database application programs to this database system and to implement additional ones. The means to input and display all applicable data is in place. The entire Form Library is in place with ample screens to encompass those original and expanded requirements. The ability to access, add and modify student and faculty personal history data is fully operational. The conversion and writing of those applications regarding student ed plan manipulations and other utilities are still pending. Student database projects completed within database classes could be utilized to add those application utilities to the database in a more timely manner.

Another recommendation is that the maximum size DBMOD permissable be determined prior to completing the database for multi-user operation, as discussed in Chapter 4. As previously discussed, only a few of the multiple work-space areas need be deleted (preliminary work indicated that any two of the variable work-space areas would suffice). It was decided not to make this change due to limited time requirements for needed comprehensive testing procedures. Those data-sets determined during this project to require multiple work-space areas and were originally specified in the DBG are listed in Appendix K.

As much as this project extolled the utility of the FMS specialized display technique, another recommendation is that AFIT obtain the

documentation for the upgraded versions of FMS. This project utilized Version 1.0 which was limited from utilizing embedded PASCAL FMS calls which Version 2.0 supports. In this manner additional use can be made of the utility by directly embedding the FMS calls within the application programs and enhancing the programmer's ability by allowing the programmer to write both the program and call forms using PASCAL alone.

The implementation of the working DBA (as discussed in Appendix N), should portend few if any problems. Acknowledging that obtaining additional personnel to care for the AFIT/ENG database may not be possible, this project emphasized that additional personnel need not be required and these tasks may be absorbed as additional responsibilities. Both positions characterized by the DBMS and DBTA presently have individuals functioning in their respective capacities. The initial question is whether the third individual will be assigned, based on the project recommendation. However this is accomplished, this position should be looked into to insure adequate and complete data records continue to be kept.

The performance characteristics of the database should also be investigated in an effort to improve performance and obtain the best operation given the available means (23 and 25). Additional work to provide this information should be done within the framework of an additional thesis effort. Work to complete this network database to obtain a true multi-user network of terminals utilizing the TOTAL multi-user approach should also invite further study.

BIBLIOGRAPHY

1. Date, C.J. An Introduction to Database Systems (Third Edition). Reading, MA: Addison-Wesley Publishing Company, 1982.

2. Allred, Dean S. Consolidated AFIT Data Base, Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1980 (AD 124376).

3. Ricks, Jeffrey S. and Robert S. Colburn. Analysis of Information Requirements and Design of the Consolidated AFIT Database and Information System (CADIS) with an AFIT/CI Implementation Design, Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1980 (ADA 124647).

4. Zelkowitz, Marvin V. "Perspectives on Software Engineering", ACM Computing Surveys, 10: 199-211 (June 1978).

5. Peters, Lawrence J. Software Design: Methods and Techniques. New York: Yourdan Press, 1981.

6. Cincom Systems, Inc. Minicomputer TOTAL Database Administrator's Guide for VAX-11 Systems. Publication No. P10-0001-02. Cincinnati, OH: Digital Equipment Corporation, 1980.

7. Cohen, Leo J. Pre-Data Base Survey. Princeton, NJ: Performance Development Corporation, 1979.

8. Ullman, Jeffrey D. Principles of Database Systems (Second Edition). Rockville, MD: Computer Science Press, 1982.

9. Bequai, August. How to Prevent Computer Crime: A Guide for Managers. New York: John Wiley and Sons, 1983.

10. Fernandez, Eduardo B. and others. Database Security and Integrity. Reading, MA: Addison-Wesley Publishing Company, 1981.

11. Jones, Paul E. Data Base Design Methodology: A Logical Framework. Wellesley, MA: Q.E.D. Information Sciences, Inc., 1976.

12. Lyon, John K. The Database Administrator. New York: John Wiley and Sons, 1976.

13. Martin, James. Computer Data-Base Organization (Second Edition). Englewood Cliffs, NJ: Prentice-Hall, Inc., 1972.

14. Wiederhold, Gio. Database Design. New York: McGraw Hill, Inc., 1977.

15. Inmon, William H. and L.J. Friedman. Design Review Methodology for a Data Base Environment. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1982.

16. Cincom Systems, Inc. TOTAL User's Guide. Digital Equipment Corporation, Canada, 1980.

17. Ralston, Anthony (Editor). Encyclopedia of Computer Science and Engineering (Second Edition). New York: Van Nostrand Reinhold Company, 1983.

18. Battelle Laboratories. An Analysis of SAMMS under TIS. Columbus, OH: Battelle, 1982.

19. Donna, Nick, TOTAL Software Developer. Personal interview. CINCOM Systems, Cincinnati, OH, 13 August 1984.

20. Digital Equipment Corporation. VAX-11 FMS Software Reference Manual. Order No. AA-J260A-TE. Digital Equipment Corporation, Maynard, MA, September 1980.

21. Digital Equipment Corporation. Reference for VAX. Digital Equipment Corporation, Maynard, MA, December 1980.

22. Digital Equipment Corporation. VAX-11 FMS Release Notes. Digital Equipment Corporation, Maynard, MA, December 1980.

23. Bailor, Paul D. Development of a Data Base Management System Performance Monitor, Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1983.

24. Milne, Robert. Lecture materials distributed in EE593, Software Engineering. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, February 1984.

25. Brunder, Timothy D. Continued Development of a Data Base Management System Performance Monitor, Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1984.

26. Perron, Robert. Design Guide for CODASYL Data Base Management Systems, Wellesley, MA: Q.E.D. Information Sciences, Inc., 1981.

27. Carroll, John M. Data Base and Computer Systems Security. Wellesley, MA: Q.E.D. Information Sciences, Inc., 1980.

28. Hoffer, Jeffrey A. Methods for Primary and Secondary Key Selection. Wellesley, MA: Q.E.D. Information Sciences, Inc., 1980.

29. Cincom Systems, Inc. TOTAL Training Notes for VAX Computers. Publication No. P10-0005-01, Digital Equipment Corporation, Cincinnati, OH, 1980.

30. Cincom Systems, Inc. Minicomputer TOTAL Programmer's Reference Manual for VAX. Publication No. P10-0002-01. Digital Equipment Corporation, Cincinnati, OH, 1979.

31. Masty, Allan F. Development of a Communications Front End Processor (FEP) for the VAX-11/780 Using an LSI 11/23, Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1983 (ADA 138152).

32. Ullman, Jeffrey D. Principles of Database Systems. Rockville, MD: Computer Science Press, 1982.

33. Lyon, John K. The Database Administrator. New York: John Wiley and Sons, Inc., 1976.

34. Pressman, Roger S. Software Engineering: A Practitioners Approach. New York: McGraw-Hill Book Company, 1982.

Vita

Myron Erich Pangman was born of a military family in Augsburg, Germany on 29 July 1950. He graduated from Texas Military Institute High School in San Antonio, Texas in 1968 and attended the United States Military Academy from which he received the degree of Bachelor of Science and was commissioned in the U.S. Army in June 1972. He served in a variety of Field Artillery and Army Aviation assignments. His last assignment was as Commander, 61st Company, U.S. Army Aviation Center, Ft. Rucker, AL. He entered the Air Force Institute of Technology in June 1983.

Permanent Address: 15266 Oak Spring
San Antonio, Texas 78232

Appendix A

Data Questionnaire to Complete Development and Implement AFIT/ENG Database

The purpose of this questionnaire is to gather data to assist in completing development and implement the AFIT/ENG Database management system utilizing TOTAL. Proposed implementations encompass the Graduate Student Credit Record and the Faculty entries. This portion of the database will exist on the Digital Lab VAX 11/780 as does the currently running Student portion of the AFIT Database. Your answers will provide the input for the system composition and determine how well it may assist your office and work.

This interview form is designed to be used in conjunction with personal interviews of those selected individuals most likely to have direct input into the overall design and implementation of the AFIT/ENG Database. It is understood that this interview form is not in and of itself complete and all inclusive.

The accuracy and completeness with which you respond to the following and any follow-on questions which may arise will determine the outcome and content of this portion of the AFIT/ENG DataBase.

1. What kind of data is required by your organization? (Can you catagorize this data within such areas as Faculty, Student, Graduate Student Credit Record, organization, etc.?)
2. Can you further define each major area and discuss functions and procedures which relate to and among each other?
3. What attributes do you require for each piece of data?
4. What are the attribute names, values and sizes for each piece of data?

5. Briefly explain any relationships among your data.

6. Briefly explain any relationships within each data.

7. Are there any queries you might routinely search for in the Database?

8. What is the format in which queries should appear?

9. How do you envision the output format should appear?

10. How user friendly do you envision this portion of the AFIT Database? (i.e., how computer query literate will be those users who utilize this portion of the Database?)

This work is being conducted by MAJ Mike Pangman in the pursuit of satisfying requirements to obtain a Master of Science degree. Thank you for your assistance.

Appendix B

Interview Subject Responses

The following lists the interview responses conducted to anticipate future and present user database requirements. The listing is subdivided by subject areas and contains those requirements the interviewees requested to have incorporated into the expanded AFIT database. Duplications are preserved to display the full interview responses.

STUDENT

1. Presently, updating is tedious:

   - Can only change one field during one operation, would prefer doing all required changes on a record during one operation;

   - Presently, when searching for a name, the search returns every name prior to presenting a listing of the names which match the request. At this point the number from the list corresponding to the desired name is chosen to obtain the listing. However, if the incorrect name is chosen, the process must be repeated rather than have the previous sort retained and available for a second try.

2. Add the Faculty Advisor to the database.

3. List the Instructors and courses each teaches or is scheduled to teach:

   - Be able to query instructor schedule for future courses;

   - Query for a particular instructor or course in a particular quarter.

4. Wants to PERSONALLY get listing of number of students enrolled for each course for future, without requiring DBTA to do it for him:

   - Would like to get notice printed out for each individual signed-up for a course when it will not be offered - like STUDENT-SECTION DESIGNATION-ADVISOR.

5. Currently gets a printout containing Student/quarter/courses/hours/grades; would like abbreviated version also.

6. Would like Independent Study (Thesis/Doctorate) course to represent exactly how many hours are scheduled for each

quarter - do it for 1-12 hours; change default from current 4 hours to accommodate this.

ADVISOR

1. Revise the database to encorporate a management tool which can determine GPAs and assist in filling out Graduate Record data for requisite forms. This would be most helpful - especially since this is needed quickly just prior to Graduation:

   - calculate GPA for 48 Graduate Credit hours;

   - determine remaining courses and calculate their GPA;

   - calculate a GPA achieved by all the Graduate Courses (minus all the undergraduate courses);

   - calculate GPA consisting of all courses taken.

2. Be able to supply data for AFIT form 78/79(?).

3. (A few instructors were not interested in personally inputing or retrieving this data, as long as DBTA or secretary provided this support.)

4. Course Scheduling should be projected out to 18 months to take advantage of building Student Ed Plans. Since Advisor is responsible for maintaining current Ed Plan of his students, he should be able to make updates to the database, while others, who do not have a need, are kept out. An anticipated reduction of paper flow and differing versions of Ed Plans between the Department and AFIT Administration are expected with this database expansion. Database could also be used as a means of recording coordination of Ed Plans or other items made between individuals. The advisor could then give each student a copy of his Ed Plan. (EMPHASIZED AGAIN) This would assist the class advisor, individual student advisor and thesis advisor more quickly than present system.

5. System should also handle incoming students data prior to their actual arrival which would necessitate being able to insert partial data.

6. Enter end-of-course grades on line by instructor to flesh out Ed Plan; Registrar would still receive a copy of grades which could reduce the amount of hardcopies shuttled back and forth. Electronic signature to verify this type of data (and data transfer) is a future possibility.

   - Security - This would be of great concern for an on-line Ed Plan.

- Entering individual data requires Social Security Number (SSN) or a sort routine as exists to choose among students by name thereby bypassing the security the Social Security Number (SSN) provides.

- Grades in database also create a security access problem.

- Update Ed Plan using separate grades file either automatically or have it done separately when updating Ed Plan.

MANAGEMENT TOOLS

GRADUATE CREDIT RECORD

1. Used as official document for approval or disapproval of all graduations.

2. Ability to calculate GPA's -- presently it must be accomplished by hand prior to Standards Committee meeting and approving graduates for graduation.

3. Need database to split course hours by number of design and theory hours AND have this break-out listed properly onto Graduate paperwork which requires it broken out by design and theory hours. (One 4-hour course broken into 2 hours credit to each area.)

ED PLAN

1. Access available on individual terminals.

2. Ed plan output format should confirm to AFIT Form 29 Ed Plan. Also, a routine to check a tentative Ed Plan to determine whether the data contained within it is valid. Those which do not match existing requirements have a 'flag' set so that errors may be displayed after the check has been made. (e.g. a '4' level course could have separate 'validated' or 'not validated' items)

3. A routine to determine whether listed courses in Ed Plan are in fact offered for that particular quarter.

4. A routine to DELETE entire Ed Plan when required rather than having to delete each item (course, quarter, etc.), including student name and Social Security Number (SSN) - the inability of the present database to do this adds time in order to eliminate data piecemeal.

5. A routine to enter NEW COURSES, SECTIONS and QUARTERS into database as required by the secretary, rather than wait for DBA to enter them.

- Would like to obtain, on a calender or academic year basis (and probably by quarter as well), the teaching load sum by individual
- Would also like to obtain Professional Development Quarter listing; list should be broken out by quarter (this will alert him as to why his load is low, or which quarter his instructor is 'free')

9. (*) Graduate Record Data

- Verify Degree Requirements
  * check sequence entries (know which courses make up sequences)
  * Proper # math credits not in sequences and not programming courses (would need to list individual courses or code math programming courses)
  * Determine sequences: at least 7 hours/sequences but total of 2 sequences must equal minimum of 17 hours
  * Other Graduate level courses - minimum total equals 48 hours (check to insure ONLY Graduate courses up to here & NO RESTRICTED COURSES <Restricted courses from 48 hour total for Graduate Credit: EE 545, EE 560, EE 562, EE 572, EE 573, any Systems Management (to include Operational Sciences also) and CT courses>)

* At the point where the teaching load is determined, he wants to be able to determine and input to the Instructor Load Determining Function the number of sections, based on the known number of students signed up for the course - does not want to rely on a magic figure to split a course into subsequent subclasses (15 is the usual figure for dividing labs) — If Instructors share a course, determine load and divide the total equally among instructors (2 or more)
* For lecture portion of the course, use 30 as a default figure - but still have option to break up a smaller class into smaller sections (be able to change default option?)
* Load also includes number of theses instructor handles - computed as 18 contact hours per completed thesis by completion date - would like to query on academic (fiscal) year (FALL - SUMMER), and calender year (WINTER - FALL).
* Load also includes any short term teaching hours - this is computed on the ACTUAL NUMBER of Hours person teaches
* Interested in obtaining the total load time and not the breakout by lecture and lab

7. (*) Student Ed Plan

   - List name & quarter & course when scheduled course not offered
   - Able to flag errors on Ed Plan
   - Accept grades
   - Flag prerequuisites - need to know what they are; flag if not met or waived.

8. (*) Determine Teaching Loads - would like to get individual or entire list of faculty teaching loads (MANAGEMENT TOOLS)

   - Teaching Loads defined as:
     * If a lecture course - multiply # lecture hours by 11
     * If a lab course, multiply the Lab section time by 11 (# of actual hours of time spent a week in lab)
     * EXAMPLE: 4 Hours total course broken out by 3 hours lecture, and 3 hours lab (called direct teaching hours); 3 lecture hours x 11 = 33 + 3 lab hours X 11 = 33 also, for a total of 66. (normal credit hours are 3 for lecture and 1 for lab, but actual # of lab hours is 3)
     * If two lab sections, then the number for that is counted twice (total for 2 labs would then become 99)

- all students within each class
- faculty advisors
- thesis advisors
- thesis data - STUDENT/ADVISOR/BOX #/CLASS/THESIS TOPIC/COMMITTEE MEMBERS

4. Internal Course Data - STUDENT/HOME PHONE #, BOX #, PROJECT ASSIGNED
5. (*) Army Students class data - arranged so they are counted in Database
   - arranged by Fiscal & Academic year
   - arranged by class designation
   - determine class leader & faculty advisor
6. (*) Instructors linked to courses (past, present, future)
   - Under Course - TITLE, # CREDIT HOURS, TEXT USED, INSTRUCTOR, QUARTER
   - Determines projected enrollments
   - Used to determine whether a class will be taught or not based on number of students signed up through Ed Plans; this could be projected as early as first quarter after completing Ed Plans
   - Useful for notifying students if a course is to be cancelled - to alert them and advisor to revise Ed Plans
   - Useful for determining number of books to be ordered (unsure as to who really determines the number of ordered texts)
   - To assist instructor book ordering and room scheduling -- Subject, Number of Students Scheduled Per Quarter.

Appendix D

Prioritized Listing of Proposed New Database Requirements

The subject areas listed in this Appendix were extracted from the interviews conducted to achieve responses to the areas required to be addressed by the results of this project. Each area has a designation as to the priority for its undertaking. This designation is indicated by asterisks next to the numbered subject area - one asterick is for first priority, two asterisks are for second priority, and three asterisks are for third priority.

1. (*) Thesis Course - revise default credit for thesis course from 4 to any number of hours

2. (*) Improve menus in database
   - more user friendliness
   - provide more help to navigate through system
   - be able to use lower case letters
   - correct present menu
     * 'L' or 'S' is asked for and proper response not received since program is looking for a numerical response

3. (*) Additional Links:

   Faculty Advisor --->students
   Faculty Advisor --->thesis students
   student --->faculty advisor
   student --->thesis advisor

   Link to class:

{Course Data: COURSE NUMBER, CREDIT HOURS, LECHRS, LABHRS,
DESCRIPTION, OUTLINE, SIZELMT}

Professional development quarter: FSSN, QUARTER

{Rooms: ROOM, BLDG, NUMBER ALLOWED

Schedule: DATE, TIME, ROOM, COURSE NUMBER}

Ed Plan: COURSE, SEQUENCE, PREREQ, SSSN, SECTION, GRADES, FSSN

Thesis Committee Members: FSSN, Thesis Number, SSSN

Sequence: Sequence name

{Prereq:} COURSE NUMBER, PREREQ, NUMBER

{Offerings:} COURSE NUMBER, QUARTER

{Dates:} QUARTER, DATE

Degree Req: THESIS, MATH, SEQUENCE, GRADES, TITLE

Restricted Courses: COURSE NUMBER, TITLE

{TITLE: COURSE NUMBER, TITLE}

Appendix C

Attribute Listing

This listing compares the records and attributes listed in Allred's thesis (2:55) with those atomic data values initially determined after the interview process was completed. Those items listed in { } are the items envisioned for the AFIT database using the Relational scheme with the keys for each record underlined. As depicted, a number of keys are useful in the TOTAL network model. This information is strictly for comparison between the original relational design scheme and the established AFIT database using a network DBMS.

Faculty Professional Data {(Committees): FSSN, Committee}

(Army) {Students: SSSN, Name, rank, PAFSC, DAFSC, Ed Code, DOR, DOC, DOB, POB, Phone, Duty Phone, Address, Emergency Address, Spouse, Spouse Date of Birth, Number of Dependents, Service, Race, Sex, Religion, Aero Rating, Losing Command, Military Spouse, Years of Service, Last Organization, Position Title, Duration}, Graduated AFIT (Y/N)?

Class: STUDENT NAME, SSN, BOX #, PHONE #, CLASS LEADER, FACULTY ADVISOR, THESIS ADVISOR

{Thesis Data: THESIS NUMBER, TITLE, SPONSOR, LOCATION, CLASSIFICATION}, FSSN (ADVISOR), THESIS COMMITTEE MEMBERS

Faculty Advisor: FSSN, STUDENT NAME, SSSN

Thesis Advisor: FSSN, STUDENT NAME, SSSN, THESIS #

Class Leader: SSSN, CLASS

Instructor statistics: FSSN, COURSE NUMBER, QUARTER, SSSN, GRADE, TITLE, TEXT USED

{Grades: SSSN, COURSE NUMBER, QUARTER, GRADE}, COURSE DATA

Teaching Load: FSSN, DEPT, THESIS NUMBER, COURSE DATA, PROFESSIONAL DEVELOPMENT Quarter.

- Total number Students in each Class given quarter, year, section.

4. CLASS GROUP LISTINGS - like CST84J. Associate these groups by Fiscal Year with the number of students in each class, the class leader of each group, and the faculty advisor for that class. Add Army classes to database to allow integration with other students' Ed Plans to assist instructor in projecting class sizes, books, projects, etc., better.

5. Link all the individuals within a class, faculty advisor for each student, and each student's thesis advisor.

   Faculty Advisor --->students
   Faculty Advisor --->thesis students
   student --->faculty advisor
   student --->thesis advisor

6. Link current courses and past courses taught with final grades for each student for those courses.

7. Assist instructor book ordering and room scheduling -- Subject, Number of students scheduled per quarter.

8. Display student box number, phone number, advisor, class designation.

9. Keep track of student already graduated to provide data for forwarding materials or to get hold of him if needed - for security clearances.

10. As it stands now, only very few individuals have access. Therefore, 'users' do not know what is on line and what they can have access to.

11. Need application programs - determine the highest need, even if it is already there, and get those operational and accessible.

SECURITY

1. Faculty advisor should be able to update student Ed Plans and students should be able to call up their classes, Ed Plans, and grades.

2. Database is only as good as the data entered into it.

3. From a query viewpoint, should be able to get data from terminal in office; to change data, should have to go to find administrator to get approval.

4. Security is of paramount concern.

- NAME, COMPANY, ADDRESS, INTEREST IDENTIFIERS IDENTIFIED BY USER, DATE OF EXPIRATION OF HIS TERM ON A COMMITTEE, DATE DATA ENTERED OR MODIFIED, MISC., RESPONSE FIELD, FIELD TELLING IF DATA HAS BEEN SENT.

3. Cataloging invited attendees for a conference or workshop.

4. Cataloging authors who submit papers or abstracts and the papers written (title, date submitted, reviewed, published, yet published, refereed or non-refereed journal).

5. Keep track of any consulting done (name of organization and contact at organization; and how correspondence was accomplished - by telephone, letter, or how).

6. Meetings attended, talks/briefings (where given, title, date given).

7. Research Proposals written and submitted:

   - title, date submitted, results of proposals, members of proposal, subject, amount of money requested.

8. Acquisition of Computer Equipment - AFIT ACD; maintaining lab equipment requests - cost, submitting requests.

9. Everything listed relational within a major area (ex., Determine students in GCS-84D taking course whom instructor has not yet seen).

OTHER/REQUIRED KINDS OF DATA

Determine:

1. Number of students in a given course in a given quarter.

2. Information on each Student's Ed Plan

   - Form 29;

   - Form 89;

   - Form 78.

3. Format of information:

   - Instructor, quarter, Year, Course, Section number;

   - Student Identification, Class, quarter, Year, Section number;

   - THESIS INFO - Student, thesis number, title;

- Obtain Professional Development Quarter listing; this should be broken out by quarter (this will alert user as to why individual instructor load is low, or which quarter instructor is 'free').

- Need to catalog faculty publications and instructor consulting.

- Top priority should be to get Student Database upgraded; this will off-load the faculty work load and be more beneficial than having a Faculty database.

- Would like to know who is projected to teach courses as far in advance as possible (see page 2). Under the course -- should have -- title, no. of credit hours, text, instructor, quarter matched up with the instructor, and double check with all students' Ed Plan to insure no duplication of students within courses. At least know what the enrollment for a course is.

- Room usage could be useful -- using the big chart ENA has in cases where class gets rescheduled. Required information: number of students room can hold, hour each room is used. Scheduling conflicts become a problem when instructor has to teach 2-3 sections of same class when each section has uneven number of students in them. The need to know the number of students in each course is imperative.

- Instructor wanted to know if the current system is able to project enrollments -- that is, a teacher should be able to query the database and find how many students are enrolled for a particular course. This assists the instructor in ordering the number of texts needed for his class. Orders for books are submitted 2-3 months in advance. Instructor determines the book used with a course and the academic department provides a catalog number that corresponds to the ordered book. (Perhaps some record between the name of the text and the number the department gives it should be recorded?)

PROFESSIONAL = FACULTY

The database should keep track of:

1. Professional duties (like chairman of an IEEE committee).

2. Any mailing list an individual in this situation has responsibility for - Attributes could resemble this:

- Teaching Loads defined as:

  * If a lecture course - multiply number of hours by 11.

  * If a lab course, multiply the Lab section hours by 11 (number of actual hours of time spent a week in lab).

  * EXAMPLE: 4 hour course broken out as 3 hours lecture and 3 hours lab (called 'direct teaching hours'); 3 lecture hours X 11 = 33 plus 3 lab hours X 11 = 33 also, for a total of 66. (normal credit hours are 3 for lecture and 1 for lab, but actual number of lab hours is 3).

  * If there are two lab sections, then the total for one is counted twice (total for 2 labs would then become 99).

  * Must reserve ability to determine and input to the Instructor Load Determining Function the number of sections, based on the known number of students signed up for the course - does not want to rely on a magic figure to split a course into subsequent sub-classes (15 is the usual figure for dividing labs) -- If instructors share a course, determine load and divide the total equally among instructors (2 or more) *** Might consider having the program determine the number of sections based on 15 per section and the number of students signed up, but prior to computing, ask if the user concurs with the computed number of sections to be used to determine the instructor load data - if not, user determines number of sections on which to compute data.

  * For lecture portion of the course, use 30 students per class as a default figure - but still have option to change default option.

  * Load also includes number of theses instructor handles - computed as 18 contact hours per completed thesis by completion date - would like to determine this data by academic (fiscal) year - (FALL - SUMMER), as well as by calender year (WINTER - FALL).

  * Load also includes any short term teaching hours - this is computed on the ACTUAL NUMBER OF HOURS person teaches, not the course credit assigned.

- Obtain on calendar or academic year basis (and probably by quarter as well) the teaching load sum by individual.

(student grade data could be kept within the database under this circumstance).

7. Other instructors want individual access to data from terminals within their offices - highly interactive, yet protected (SECURITY).

8. Able to query and print-out from the Database which courses students have scheduled for any quarter.

9. Would like to know students signed up for a future course:

   - Used to determine whether a class will be taught based on number of students signed up through their Ed Plans; this could be projected as early as first quarter after completing Ed Plans (see page 3.)

   - Useful for notifying students if a course is to be cancelled - to alert them and advisor to revise Ed Plans.

   - Useful for determining number of books to be ordered (unsure as to who really determines the number of ordered texts).

   - To assist instructor book ordering and room scheduling -- Subj, Number of Students Scheduled Per Quarter.

USER FRIENDLY

1. An interactive system to list any discrepancies within a student Ed Plan to alert the advisor of possible problems. Flags inconsistencies and displays them at the end of the automated 'checker' for the user.

2. Present database is difficult to work with and does not provide sufficient 'help' to quickly navigate within system.

3. Lower case letters are not recognized in areas.

4. The menu does not follow itself (it asks for a choice of 'L' or 'S', for example, when it really wants a number; it does not tell you when the disk which contains the database is not accessable).

FACULTY

1. Teaching Loads - obtain individual or entire list of faculty teaching loads (MANAGEMENT TOOL section).

* Recognize restricted courses from 48 hour total: EE 545, EE 560, EE 562, EE 572, EE 573, any Systems Management (to include Operational Sciences also) and CT courses;

* Determine FCR GPA's: (See page 2);

OTHER TOOLS

1. Determine Future Class Sizes and thus determine instructor work load manpower requirements; archive ability.

2. Thesis data by student consisting of - committe members, title.

3. Class data - home telephone numbers, box numbers, all data by COURSE.

4. By project within a class for team reports - NAME, Social Security Number (SSN), SERVICE, RANK, BOX NUMBER, HOME TELEPHONE NUMBER, CLASS (GCS-84D, etc.).

5. Compilation of statistics and data on classes an instructor has taught (when taught, average grades, history of students who were in classes) to serve as an instructor grading practices history:

   - Put all grades in computer for classwork; keep track of exams students take, weight of each exam, homework grades, pop quiz grades, project grades and weight of each toward overall grade; produce averages - for a given grade, student, quarter and overall grade.

   - Pattern grading program after 'Class' program many instructors use now. Security is an issue here, especially if students are going to be allowed access.

   - A means of producing a summary of student's course grade to date, with standard deviation per exam, pertinent data per exam with listed histogram of exam and standing, perhaps, within class; this is done for each student throughout course.

   - Ability to search for students who are doing poorly (below a certain avg) in course.

   - Menu driven system to help user with on-line help facility.

6. Interaction - Restrict the number of individuals with access to the database to Secretaries or selected individuals only

6. A routine to CHANGE any personal history Data like RANK within each master data-set.

7. An enforcement policy to keep track of who has/has not updated Ed Plan -- advisor, student with date.

8. Instructors should be able to input final course grades which automatically updates Ed Plan.

9. Some example queries:

    - What is student's running GPA at any given point - some instructors need this for undergraduate requirements.

10. Ability to find student, his advisor, box number, class he belongs to, thesis topic, who is on thesis committee, sequences he is taking -- some of this data is not within Ed Plan form.

11. Able to flag prerequisites - those not met or waived.

12. Verifies all degree requirements: Check sequence entries on Ed Plan to insure all required courses are entered for a particular sequence -

    - Devise management tool to check all entries on Graduate Credit Record form (AFIT form 89);

        * ability to tag courses for proper placement on official forms -- broken out by sequences, if appropriate;

          Like:

        * Thesis listed with 12 credit hours;

        * Math credits composed of 2 approved graduate courses which are not programming math courses;

        * Sequence courses, hours and grades (at least 7 hours per sequence, but both must equal minimum of 17);

        * Other graduate level courses (so that their total is 48-52 hours; based on adding all graduate courses to that line);

        * All other courses (CT 6.XX, undergraduate, and EE 6.98, etc.; there is a list in Department handout) - so that total of all courses is between 70 - 76(?);

* Listing of other courses - total min. of 70 hours (?)

* Check ALL against LEGAL courses - checks for spelling errors

- Faculty Advisor should be able to update Ed Plan
- System able to flag inconsistencies & list at end of automated 'checker' for user to utilize

10. (*) Instructor Statistics
    - What courses taught
    - Which quarter taught
    - Students taught
    - Grades
11. (**) Room usage program - to assist ENA planning quarter and use during quarter
12. (*) Security/Integrity
    - Instructor wants access to data from his terminal - highly interactive, yet protected (SECURITY)
    - Grades
    - Social Security Number
    - Any other data covered by Privacy Act
13. (***) Course Grading
    - Determine by Student: Grade by weight for each exam, homework, project, quiz, Avg. Grade, STUDENT DEVIATION, Histogram, Standing within course
    - Ability to query for particular student in a course OR those below a particular average

14. (**) Professional Duties

- Committee chairman, Organization Officer, etc.

15. Acquisition of Computer Equipment

Appendix E

Original AFITDB Data-set Items

These are the Data-sets of the original AFIT Database (this file found in AFITDB.DBG). Shown are the data-set name and each data-item contained within it. Each data-item is listed in the order it appears in its data-set within the Data Base Generation (AFITDB.DBG) document. Within some data-sets there are obvious duplications and some data-items are listed by their four letter abbreviations only, since there is no documentation to further define what the field was intended to contain.

MASTER FACULTY FILE

FACT - Faculty Social Security Number
- Marital Status
- Name
- Civil Service grade
- MBRH (no specification)
- Office room number
- Address
- Office phone number
- Home phone number
- Date of Birth
- Name of Spouse
- Mil/Civ rank
- Date hired
- Air Force Specialty Code
- Salary
- RDAT (no specification)
- RSTA (no specification)

MASTER STUDENT FILE

STDT - Student Social Security Number
- Sex
- Name
- Mil/Civ rank
- Box number
- Duty Air Force Specialty Code
- Primary Air Force Specialty Code
- Home phone number
- Duty phone number

- Education code
- Date of rank
- Date of commission
- Date of birth
- Place of birth
- Current address
- Emergency address
- Spouse first name
- Spouse date of birth
- Number of dependents
- Military Service
- Race
- Religion
- Aero rating
- Losing command
- Military Spouse
- Years of Service
- Last Organization
- Last position title
- Duration of last duty assignment

MASTER SECTION NUMBER FILE

SECT - Student class section designation
- Section Leader Social Security Number
- Section graduation date
- Section entry date
- Section Leader Social Security Number

MASTER COURSE FILE

MCRS - Course number
- Course credit hours data
- Course lecture hours data
- Course lab hour data
- Size limit data
- Title data

MASTER QUARTER FILE

MQTR - Quarter number
- Quarter start date
- Quarter stop date

VARIABLE SECTION LEADER FILE

SECL - (Links only)

VARIABLE QUARTER FILE

VCQR - Code 'QC' (no specification)
- Coded 'QS' (no specification)

Appendix F

Data-set Requirements Tabulations

The left side of this appendix lists the specific requirements requested that the expanded AFIT Database accommodate, while the right side shows the subsequent location where the requirement has been included within it. The data requirements obtained by 'interviews' also consisted of the repeat conversations with administrative personnel as well as those originally interviewed. This document was used to check the requirements requested against the atomic values needed to support those requests. Those requirements items which have more than one data-set listed as the location where the requirement is found indicate that either area could contain such data depending on the application program or that the application program required to obtain such data can do so utilizing the listed data-set links. This data is contained in file Appendix.F;1.

REQUIREMENT DATA - OBTAINED FROM INTERVIEWS	TABLE DATA - WHERE THE REQUIREMENTS DATA ARE LISTED IN THE DATA-SETS
1. Change the thesis default course credit from 4 to any number of hours	MCRS - Course control Number has six digits (EX. EE799A - for 1 credit hr; EE799B - for 2 credit hrs.)
2. Faculty Advisor -->Students Faculty Advisor -->Thesis students Student -->Faculty advisor Student -->Thesis advisor	FADV - Linked to FACT, STDT, SECT TADV - Linked to FACT, STDT, THES
3. Thesis Data -->Student	STDT
Advisor	TADV
Box Number	STDT
Class	SECT
Thesis Topic (Title?)	THTL
Committee Members	TCMF
Thesis Title	THTL
4. Instructors with Course Listings -->	
Instructors	FACT
Course Listing	MCRS

5.	Course Listing -->	Title	MCRS
		Course Number	MCRS
		Number Credit Hours	MCRS
		Quarter	MQTR
		Course Text	MBKT
		Instructor	VINS
6.	Projected Enrollments -->	Course	MCRS
		Number Students	SCHD
		Quarter	MQTR
		Number of Books to Order	VNMO
7.	Ed Plan -->	Student Name	STDT
		SSN	STDT
		Course (Number & Title)	MCRS
		Section (Size limit data)	MCRS
		Credit Hours	MCRS
		Class	SECT
		Quarter	MQTR
		Grades	CRSE
		Prerequisites	MREQ
		Waiver	CRSE
		Course Sequences	MSSF
8.	Teaching Loads -->	a. Instructor(s) Name(s)	FACT
		b. Course(s) Taught	MCRS/VCQR
		c. Quarter	MQTR
		d. Type of Course - Lecture/Lab	MCRS
		e. Number of Sections	**MCRS
		f. Number of Theses/ Instructor	FACT/TADV/THES
		g. Short Term Hours Taught	MCRS/FACT
		h. Professional Development Quarter Listing	VPDQ
		i. Faculty Advisor Listing	FADV
		j. Thesis Committee Member Listing	TCMF
9.	Course Data -->	Student Name	STDT
		SSN	STDT
		Service	STDT
		Rank	STDT
		Class (*e.g.*, GCS84D)	STDT/SECL/SECT
		Home Telephone Number	STDT
		Box Number	STDT

**'Size Limit' is data element

10.	Instructor Statistics -->	Instructor Name	FACT
		SSN	FACT
		Courses Taught	MCRS
		Students in Course by:	
		Name	STDT
		SSN	STDT
		Class	SECL
		Grades	CRSE
11.	Catalog Faculty Publications -->	Name	FACT
		SSN	FACT
		Title	FCOM
		Date	FCOM
		Topic (TITLE)	FCOM
12.	Room Usage -->	Room Location	BLRM
		Room Capacity	CPTY
		Time Scheduled	TIME/DAYS/SCHD
13.	Projected Enrollments -->	Course	MCRS
		Quarter	MQTR
		Student SSN	STDT
		Student Name	STDT
		Student Class	SECT
14.	Professional Duties -->	Awards	FHAW
		Committee Membership	FCMT
		Committee Duties	FCMT
		Professional Societies	FSOC
		Interest Areaa	FINT
15.	Student Data -->	Name	STDT
		SSN	STDT
		Box Number	STDT
		Home Phone Number	STDT
		Advisor	TADV
		Class	SECT
		Service	STDT
		Graduated AFIT?	STDT
16.	Graduate Record Data -->	Verify Degree Requirements	MDEG
		Course Sequences	MSSF
		Courses	MCRS
		Math Credits	MCRS
		Restricted Courses (EE545, EE560, CTXXX, Etc)	MCRS

Appendix G

AFITDB Expansion Items

The following listed data-set items were not specifically requested through the interview data-gathering procedure. The items were included within the data-sets as anticipated-for-use data items. Each set of data-items is shown with its parent data-set name listed to the left. Those items with asterisks denote new items in the database. An asterisk to the left of the data-set name indicates that the entire data-set is new. These data-elements were selected by comparing like data-sets (ex., Student and Faculty Masters) and items previously proposed for inclusion in the AFIT Database by Allred. All data-items for each data-set are listed in their entirety in AFITDB.DBG. This document is in file Appendix.G;1.

FACT - Marital Status
Mil/Civ Grade
Office Rm Number
Address
Office Phone
Home Phone
DOB
Sex
Duty AFSC
Primary AFSC
DOR
Emergency Address
Number of Dependents
Service
Race
Religion
Aero Rating
Commissioning Date
Last Organization
Last Position Title
Spouses Name
Date Hired
Expected AFIT Departure Date
Spouses DOB

*ADNR - (NTIC Number - For Thesis Cataloging)
Thesis Code

MCRS - Restricted Course

*DEPT - Department Code

STDT - Sex
Duty AFSC
Primary AFSC
Duty Phone Number
Education Code
DOR
Commissioning Date
DOB
Place of Birth
Current Address
Emergency Address
Spouses Name
Race
Religion
Aero Rating
Number of Dependents
Last Organization
Last Position Title
Last Command
Military Spouse
Graduated AFIT

SECT - Section Leader SSSN
Graduation Date
AFIT Entry Date
Number in Section

MQTR - Quarter Start Date
Quarter End Date

MBKT - Book Author Name
Book Publisher Name
Number Books Available
Price

*MORD - (Book Ordering Info)
Order number
Due Date
Company
Company Address
Company Phone Number
Company Phone Area Code

MSSF - Course Sequence Number

MDEG - Number ID Type Grad Deg

*VEDU - (Variable Education File)
Institution (University) Attended
Degree Earned
Year Degree Awarded

**This is a combination of two **previous data-sets, FEDU & **DGYR.

*VHAW - (Variable Honors and Award File)
Honor Received
Date Honor Received
Award Received
Date Award Received

**This is a combination of two **previous data-sets, FHAW & **AWAR.

FCOM - Co-author Name
Publication Date
Presentations:
Organization
Date of Presentation

*FTDY - (TDY File)
Cost of Trip
Destination
Begin Date
End Date

CRSE - Course Credit
Course Number
Course Name
Grade
Course Begin Date
Course End Date
College Attended

THTL - Thesis Sponsor
Thesis Location
Thesis Classification

VREQ - Corequisite

VTXT - Description of Book
Outline of Book

SCHD - Class Finish Time

*CLSR - Additional Space If Extra Chairs Needed
Type(s) Equipment In Room
Type of Room Code
Room Security Classification

Appendix H

Requirements Presentations

Utilizing the information from Appendices D, E, and F, anticipated application programs requirements are listed to provide additional anticipation of future database enhancements. The requirements presentation items are in the left column with the anticipated application program requirements immediately to their right. Some of the applications programs were specifically requested in the interview process while others reflect anticipated related requests. This shows the relationships between the requirements portion and the applications programs (yet to be developed). The original document is in file Appendix.H;1.

Requirements Presentations	Applications Programs
1. Teaching Loads by: 1) Individual & 2) Department	1. Show Number of short course hours taught - by individual and department 2. Determine Number of thesis committees faculty member is on. 3. Determine number of theses faculty member sponsors. 4. List professional development quarters by faculty member and quarter.
2. Ed Plan	5. Verify entered course, quarter, prerequisite and show errors (with print option) for use with other application programs. 6. Printout of Ed Plan with grades
3. Graduate Record Data (Verify degree requirements)	7. Check sequence entries for proper courses. 8. Determine GPA (4) routines

4.	Project Enrollment	9. List numbers/names of students programmed to take all classes for fiscal/calender year
		10. List numbers/names of students programmed to take one class.
5.	Instructor Statistics	11. Provide history of courses taught by individual faculty members.
		12. Provide listing of students taught by individual faculty members by course and their grades.
6.	Room Usage	13. Determine optimum room usage plan for quarter.
7.	Course Index	14. List course information (quarters course taught, instructor, text used, credit hrs, title, number)
8.	Quick-look-up of Student Pertinent Information	15. Provide info to user like name, advisor, thesis advisor, box number, home phone, duty phone, class, thesis topic, thesis committee members.
9.	New Student Data	16. Input new student data to database.

Appendix I

AFITDB Forms Library Listing

The following is a compilation of the FMS form screens contained within the library which supports the AFITDB database. Each screen must reside within a library in order to require only one opening statement within the calling routine. Failure to provide a library from which to call FMS screens would require extensive opening and closing for each form utilized by an application within the driver. The three column format is produced using the FMS form utility (FUT). The 'Form (VERSION)' column lists each form name with it's corresponding latest version number. The author added the version number heading and numbers (shown in parenthesis immediately following the 'Form' HEADING and each form name) to use as a reference when updating the library. The date column lists the latest date changes were made to each form, while column three shows the work space area, in bytes, reserved for each screen within the FMS driver work space area.

FUT Vol.1
19-Oct-84

Library DU1: [PANGMAN.MIKEDB. TOTAL] MIKELIB2.FLB;42 created: 19-Oct-84
Directory is 1 blocks long

Form (VERSION)	Date	Impure area (bytes)
TIME(1)	1-Sep-84	290
FIRST1(4)	21-Sep-84	383
VREQ(1)	1-Sep-84	307
SCHEDU(3)	7-Sep-84	369
LASFAC(3)	6-Sep-84	309
TEXTBK(4)	7-Sep-84	371
MODATA(5)	7-Sep-84	451
VMSS(2)	12-Sep-84	515
UTILIT(6)	20-Sep-84	387
STDT(13)	26-Sep-84	2791
GCR1(3)	21-Sep-84	2288
GCR2(3)	21-Sep-84	1426
CRSE(5)	19-Oct-84	977
THTL(6)	21-Sep-84	1061
FSOC(6)	21-Sep-84	514
VEDU(6)	21-Sep-84	620
VHAW(4)	21-Sep-84	712
FACT(9)	21-Sep-84	2576
FCMT(5)	21-Sep-84	788
FCOM(5)	21-Sep-84	800
FINT(4)	21-Sep-84	469
FTDY(5)	21-Sep-84	695

MBKT(4)	21-Sep-84	1017
MORD(5)	21-Sep-84	826
ROOM(5)	21-Sep-84	665
DEPT(7)	21-Sep-84	493
SECT(5)	21-Sep-84	704
CLAS(3)	21-Sep-84	704
MCRS(9)	21-Sep-84	840
MQTR(3)	21-Sep-84	543
MDEG(3)	21-Sep-84	499
STUFAC(6)	21-Sep-84	363
FACLTY(5)	21-Sep-84	453
BEGIN(2)	21-Sep-84	271
STUDNT(10)	21-Sep-84	373
MSSF(3)	21-Sep-84	490
MIKE1(4)	21-Sep-84	297
FRM29A(2)	19-Oct-84	2441
FRM29B(2)	19-Oct-84	2441
FRM29C(2)	19-Oct-84	2441
FRM29D(2)	19-Oct-84	2441

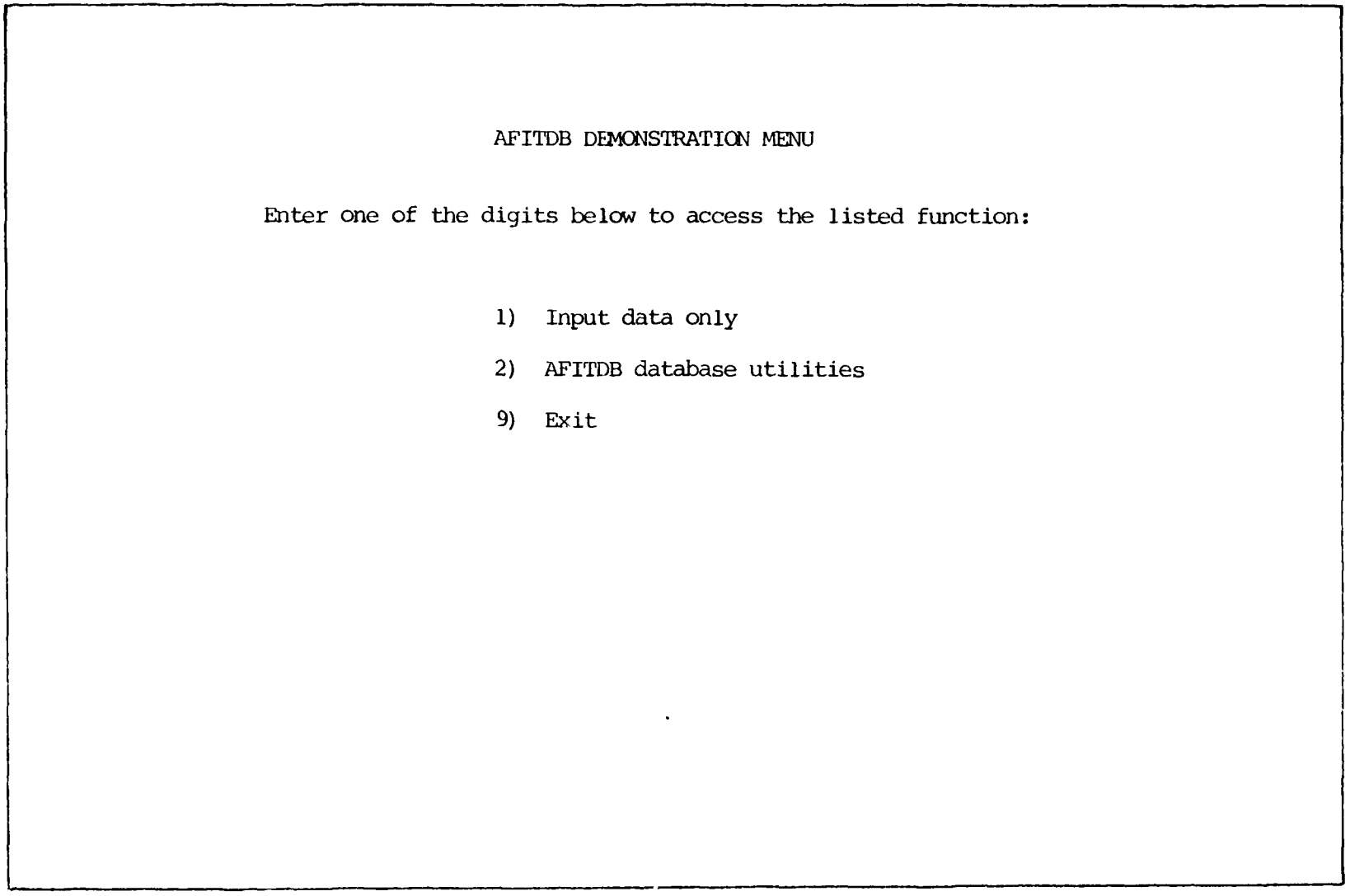


Figure I-1. Initial Form Library Screen

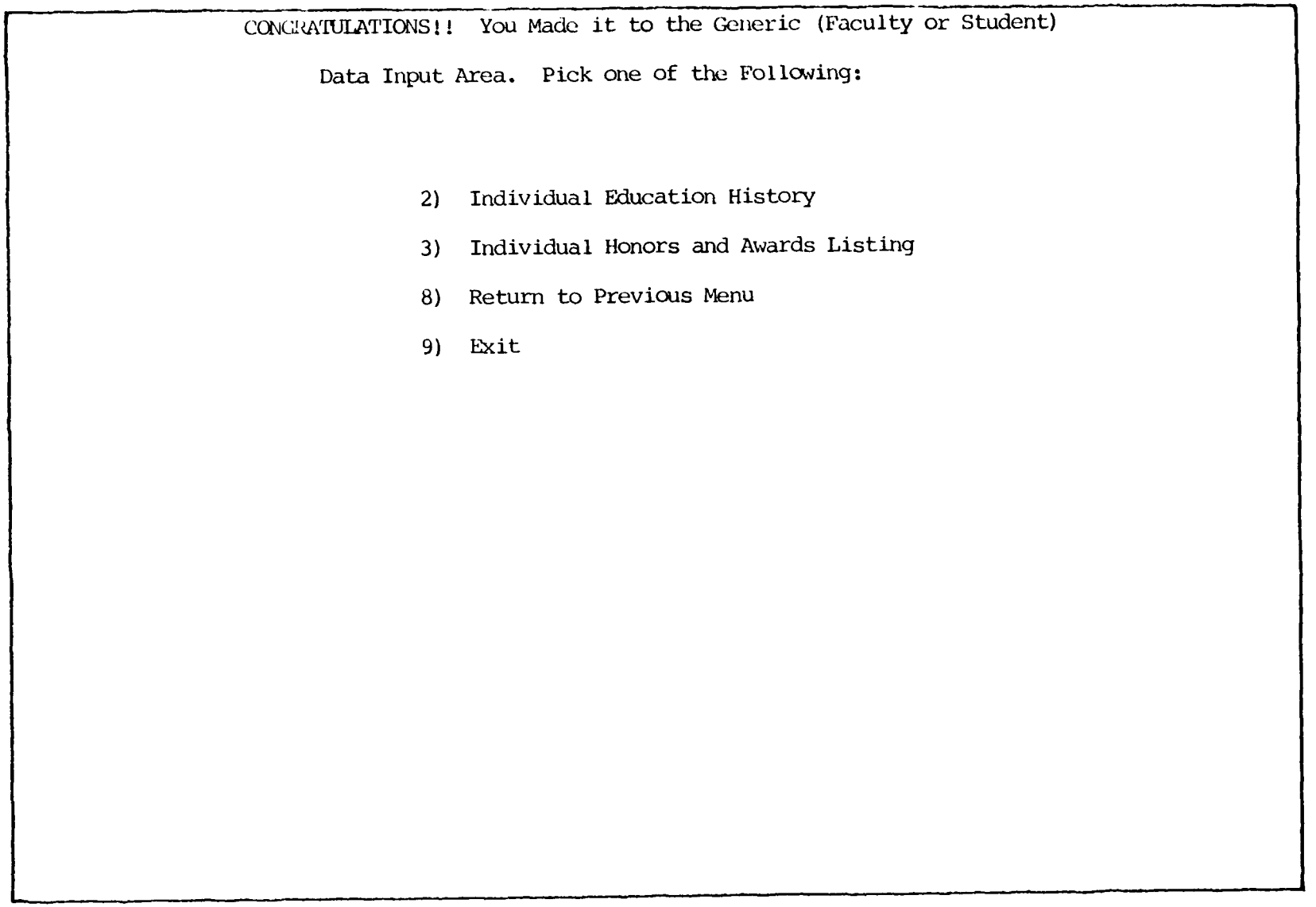


Figure I-15. 'STUFAC' Menu Selection Form

Student Thesis Data

Enter the Following Data Concerning Each Thesis:

1. Social Security Number for Author of Thesis (9): - -
2. Thesis NTIC Catalog Number (10):
3. Thesis Advisor Social Security Number (9): - -
4. Department Thesis Number (10):
5. Thesis Title (50):
6. Thesis Sponsor (50):
7. Thesis Location (50):
8. Thesis Classification (12): Unclassified
9. Do You Require Additional Entries of This form (Y/N):N

Figure I-14. Student Thesis Data

Individual Student Course Data

Enter the Following Course Data for This Student:

1. Student Social Security Number (9): - -
2. Name of Course (20):
3. Number of Course (6):
4. Number of Course Credit Hours (2):
5. Grade Received for This Course (2):
6. Quarter Student Took or Will Take This Course (4):
7. Name of School Where This Course was Taken (30):
8. Did Student Receive a Waiver Out of This Course (Y/N):N
9. Do You Require Additional Entries of This Form (Y/N):N

Figure I-13. Student Individual Course Data

Student History

Enter the Following Items:

1. Student Social Security Number (9): - -
2. Graduated AFIT (Y/N):
3. Student Name (28):
4. Box Number (4):
5. Mil/Civ Rank (3):
6. Education Code (5):
7. Military Service (2):
8. Date of Rank (6):
9. Duty AFSC (6):
10. Primary AFSC (6):
11. Date of Commissioning (6):
12. Years of Service (2):
13. Sex (1):
14. Race (2):
15. Religion (20):
16. Duty Phone (7): -
17. Aero Rating (10):
18. Current Address (40):
19. Emergency Address (40):
20. Home Phone Number (7): -
21. Date of Birth (6):
22. Place of Birth (40):
23. Marital Status (1):
24. Spouse First Name (12):
25. Spouse Date of Birth (6):
26. Military Spouse (Y/N):N
27. Number of Dependents (2):
28. Previous Command (5):
29. Duration (2):
30. Last Organization (50):
31. Last Position Title (50):
32. Do You Require Additional Entries of This Form (Y/N):

Figure I-12. Student Personal History Data

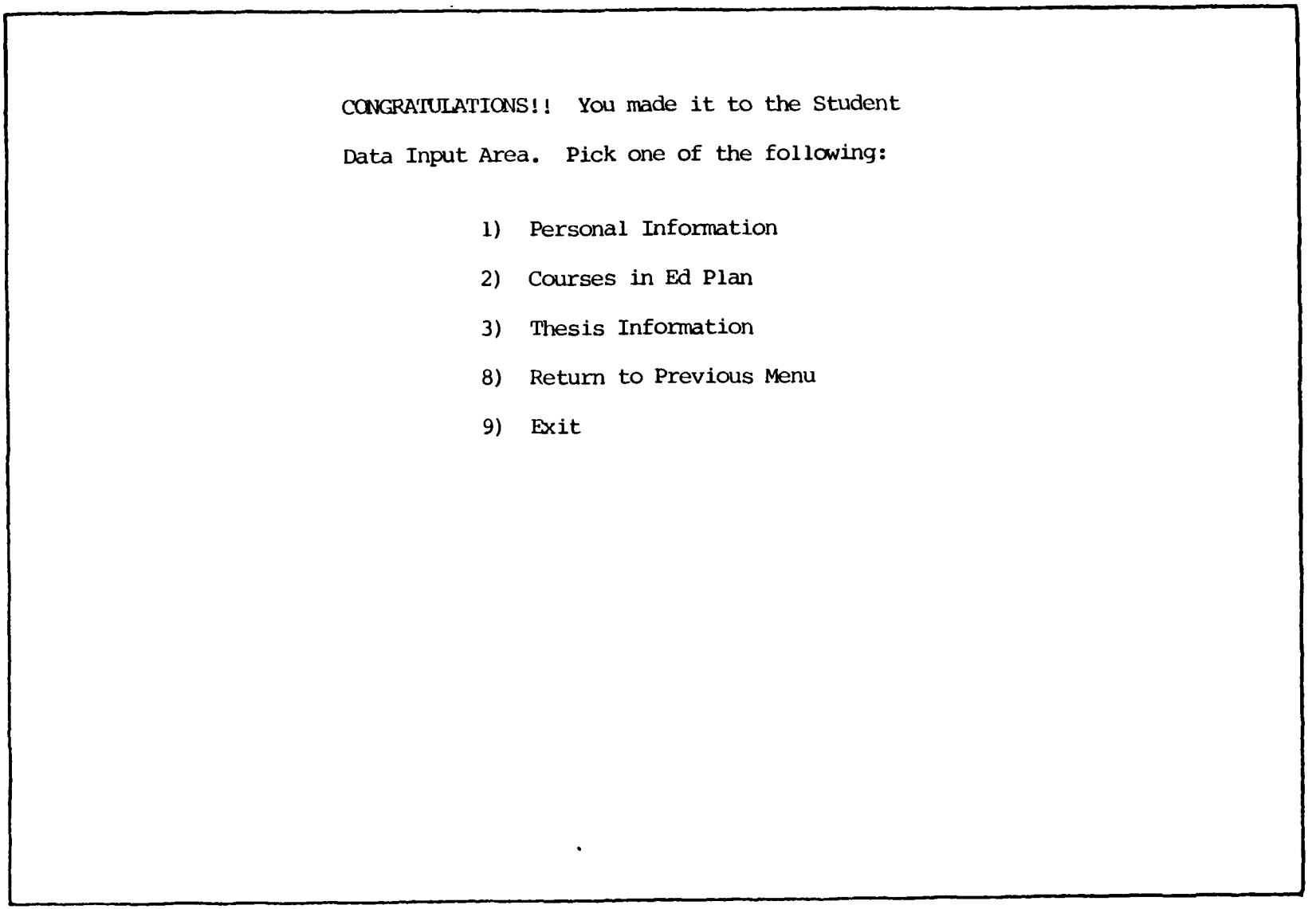


Figure I-11. 'STUDNT' Menu Selection Form

Faculty Publications and Presentations

Enter the Following Data as Listed:

1. Faculty Member Social Security Number (9): - -
2. Title of Publications (25):
3. Date of Publication (6):
4. Co-author Name (if any) (30):
5. Name of Organizations to which Faculty Member gave (or will give) Presentations (25):
6. Presentation Date (6):
7. Do You Require Additional Entries of This Form (Y/N):N

Figure I-10. Faculty Publications and Presentations Data

Faculty Departmental and Committee Memberships

1. Faculty Member Social Security Number (9): - -
2. Code for Department to which Faculty Member Belongs (4):
3. Name of Committee to which Faculty Member Assigned (10):
4. Department to which this Committee Responsible (4):
5. Additional Committee Memberships (10):
6. Department to which this Committee Responsible (4):
7. Do you Require Additional Entries of This Form (Y/N):N

Figure I-9. Faculty Committee Data

Faculty Professional Memberships

Faculty Member Social Security Number (9): - -

Name of Societies to which
Faculty Member Belongs (40):

Do you Require Additional Entries of this Form (Y/N):N

Figure I-8. Faculty Professional Membership Data

Faculty TDY History

Enter the Following Data:

1. Faculty Member Social Security Number (9): - -
2. Destination (20):
3. TDY Start Date (6):
4. TDY End Date (6):
5. Cost for This Trip (7):
6. Do you Require Additional Entries of this Form (Y/N):N

Figure I-7. Faculty TDY Data

Faculty Professional Interest

Enter the Following Data for Each Faculty Member:

1. Faculty Member Social Security Number (9): - -
2. Area of Interest (15):
3. Do You Require Additional Entries of This Form (Y/N):N

Figure I-6. Faculty Professional Interest Data

Faculty Input Data

Social Security Number (9): - -
Name(Last, First, MI) (28):
Mil/Civ Rank (3):
Military Service (2):
Duty AFSC (6):
Date of Commission (6):
Date Hired (6):
Sex (1): Race (2):
Aero Rating (10):
Office Phone (7):

Date of Rank (6):
Years of Service (2):
Primary AFSC (6):

Date of Birth (6):
Religion (2):
Salary (5):
Office Room Number (12):

Expected AFIT Departure Date (6):

Current Address (40):
Home Phone (7): -
Emergency Address (40):
Marital Status (1):
Spouse First Name (12):
Spouse Date of Birth (6):
Number of Dependents (2):
Last Organization (50):
Last Position Title (50):
Do You Require Additional Entries of This Form (Y/N):N

Figure I-5. Faculty Personal History Data

CONGRATULATIONS!! You made it to the faculty data input area. Pick one of the following:

1) Personal Information

2) Faculty Interests

3) TDY

4) Professional Societies

5) Committees to which faculty member assigned

6) Publications and Presentations

8) Return to previous menu

9) Exit

Figure I-4. 'FACLTY' Menu Selection Form

Select one digit to choose which area you desire

to enter data into AFITDB:

1) FACULTY (Personal Information, Interests, TDY, Professional Societies, Committees, Publications and Presentations)

2) STUDENT (Personal Information, Courses in Ed Plan, Thesis Information)

3) STUDENT/FACULTY (Education History, Honors/Awards)

4) TEXTBOOK INFORMATION

5) SCHEDULING INFORMATION

6) OTHER MASTER ENTRY DATA

7) UTILITIES (Ed Plan, Degree Requirements, Student Graduate Credit Record-AFIT Form 89)

9) EXIT

Figure I-3. 'FIRST1' Menu

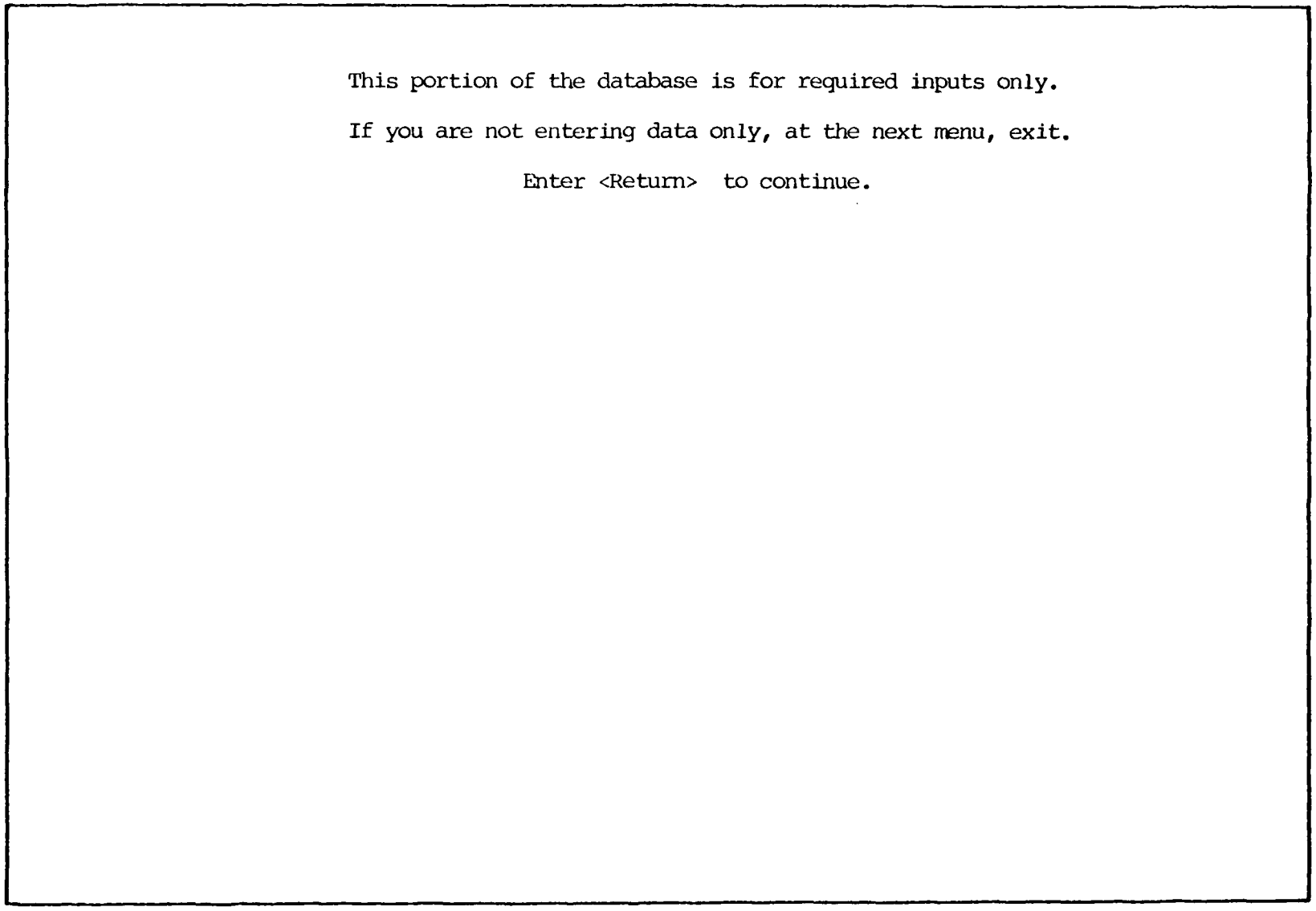


Figure I-2. Second Screen in AFITLIB2.FLB

Individual Education File

Enter the Following Information for Each Individual:

Individual Social Security Number (9): - -

Institution (University) Attended (40):

Degree Earned (40):

Year Degree Earned (4):

Do You Require Additional Entr s of This Form (Y/N):N

Figure I-16. Individual Education Data

Individual Honors and Awards Listing

1. Enter Individual's Social Security Number (9): - -
2. Honor Received (10):
3. Data Honor Received (6):
4. Award Received (10):
5. Date Award Received (6):
6. Do You Require Additional Entries of This Form (Y/N):N

Figure I-17. Individual Honors and Awards Data

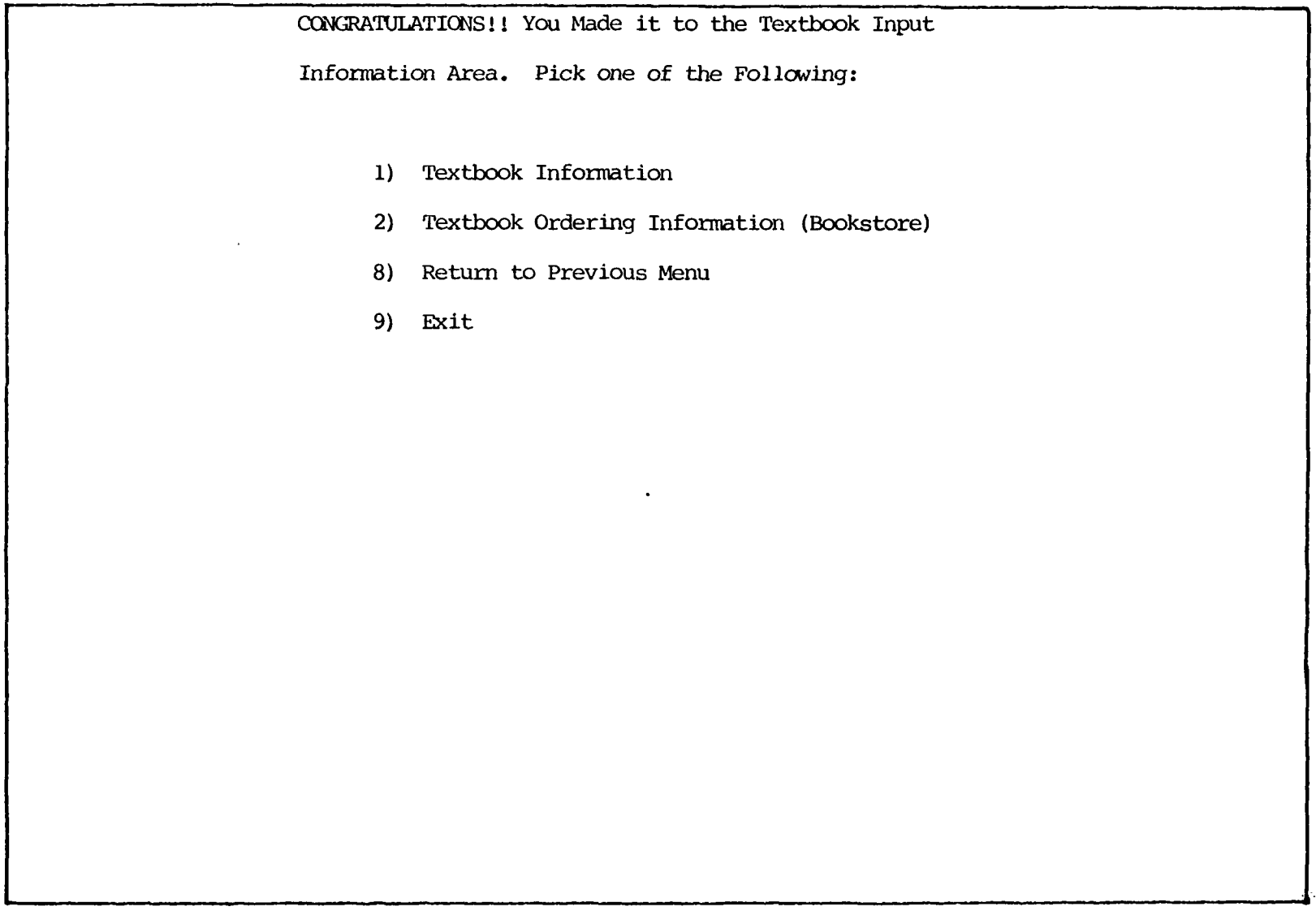


Figure I-18. Textbook Data

Textbook Ordering Information

Enter the Following Data for Each Course Text:

1. Book Title (40):
2. Author of Book Entered Above (28):
3. Publisher of Book Entered Above (28):
4. Number of This Textbook in Stock in Bookstore (6):
5. Price of This Textbook in Bookstore (40):

   Enter Textbook:
6. Description Data (50):
7. Outline Data (50):
8. Do You Require Additional Entries of This Form (Y/N):N

Figure I-19. Textbook Ordering Information

Textbook Ordering Data

Enter the Following Data Required to Order and Keep Track of Textbooks:

1. Number Designated by Department to Proposed Textbook (7):
2. Order Number (6):
3. Order Number Due Date (6):
4. Company From Which Textbooks are Ordered (20):
5. Company Address (40);
6. Company Phone Number (10):
7. Do You Require Additional Entries of This Form (Y/N):N

Figure I-20. Textbook Ordering Data

CONGRATULATIONS!! You Found the Scheduling Capability

Input Used to Match Room Sizes With Class Sizes.

Pick One of the Following:

1) Room Information

2) Allowable Class Begin Times

8) Return to Previous Menu

9) Exit

Figure I-21. 'SCHEDU' Menu Selection Form

Room Information

Enter the Following Room Data:

1. Classroom Number and its Corresponding Building Number (8):
2. What is the Student Seating Capacity for This Room (4):
3. Type of Equipment in Room (2):
4. Type of Room Code (3):
5. Security Classification Level of Room (1): U
6. Do You Require Additional Entries of This Form (Y/N):N

Figure I-22. Room Scheduling Data

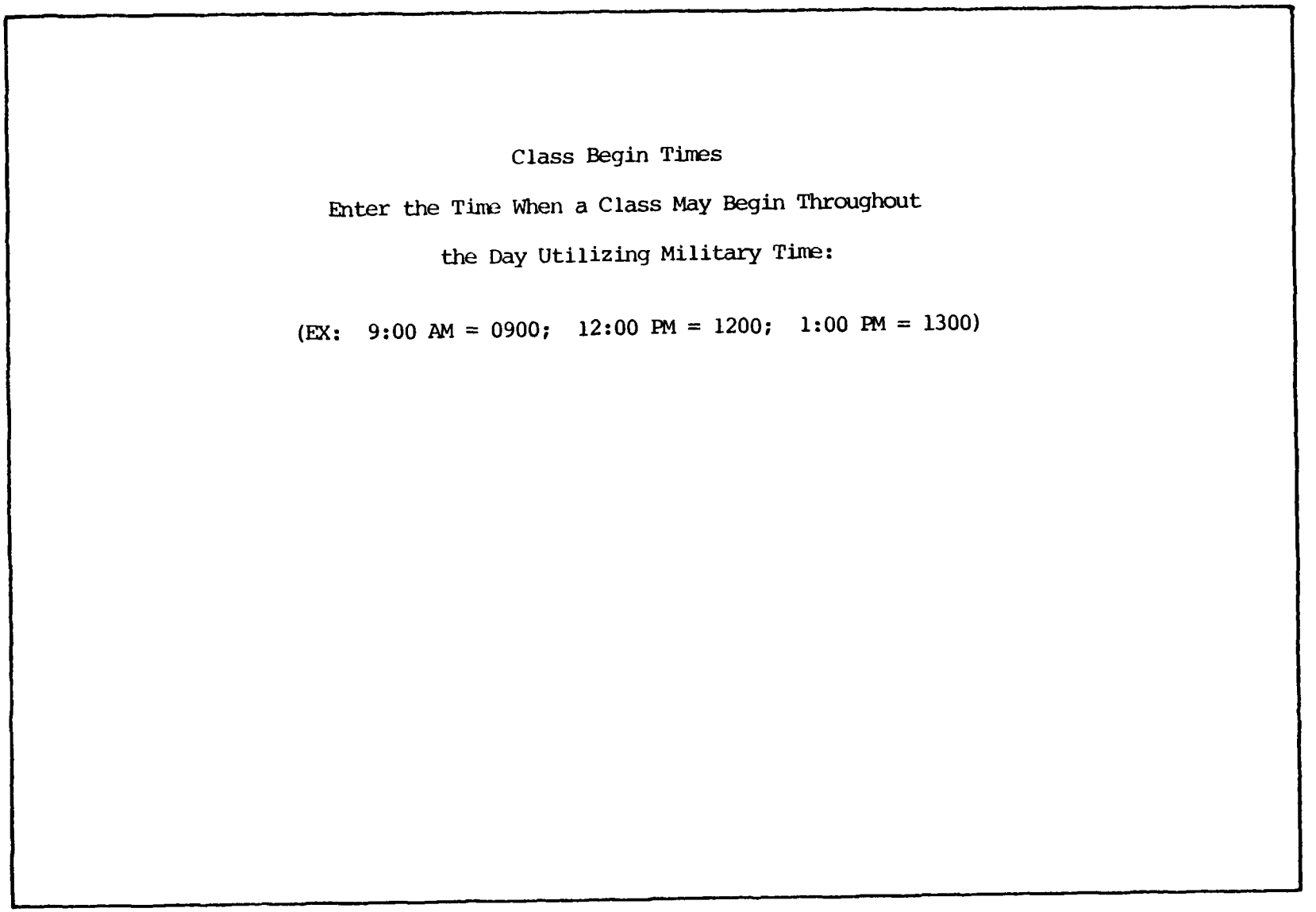


Figure I-23. Class Start Times

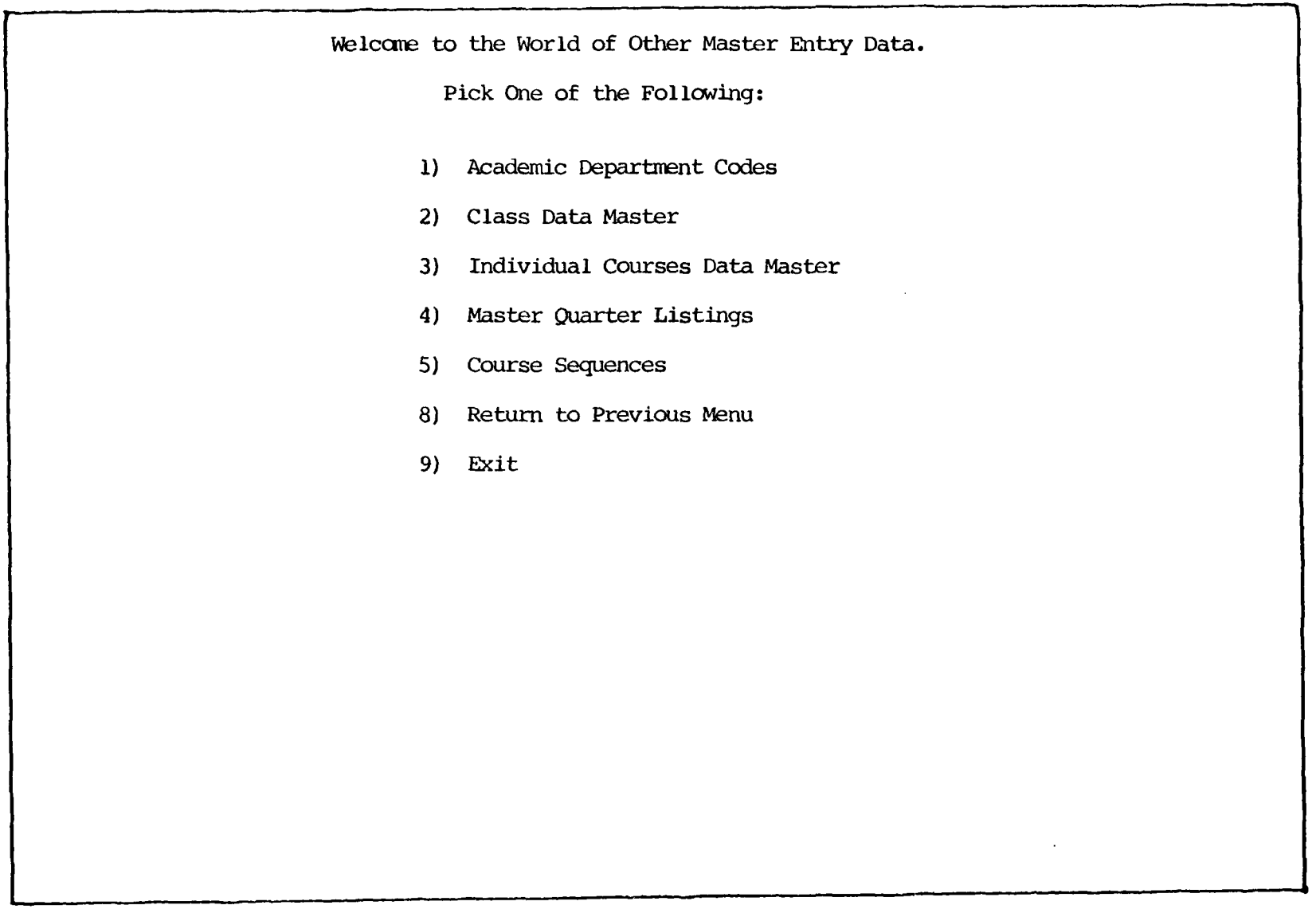


Figure I-24. 'MODATA' Menu Selection Form

Academic Department Codes

1. Enter the Academic Department Name (20):

2. Enter the Control Code You Desire To
   Match With This Department Name (4):

3. Do You Require Additional Entries of This
   Form (Y/N):N

Figure I-25. Academic Department Codes

Class Data

Enter the Following Class Data:

1. Class Designation (8):
2. Class Leader Social Security Number (9): - -
3. Scheduled Class Graduation Date (6):
4. Class Entry Date (6):
5. Number of Students in Class (3):
6. Class Faculty Advisor Social Security Number (9): - -
7. Do You Require Additional Entries of This Form (Y/N):N

Figure I-26. Student Class (Section) Data

Course Data

Enter the Following Course Data:

1. Course Number (6):

2. Name of Course (50):

3. Number of Designated Course Lecture Hours (1):

4. Number of Designated Course Lab Hours (1):

5. Planned Maximum Number of Students Per Section (2):

6. Is this Course Restricted from Use for
   Graduate Course Requirements (Y/N):Y

7. Do You Require Additional Entries of This Form (Y/N):N

Figure I-27. Separate Course Data

Enter From the List Below Whether the Course is a

Co-Requisite or Pre-Requisite (2):

Pre-Requisite (Enter 2)

Co-Requisite (Enter 1)

Figure I-28. Requisite Data

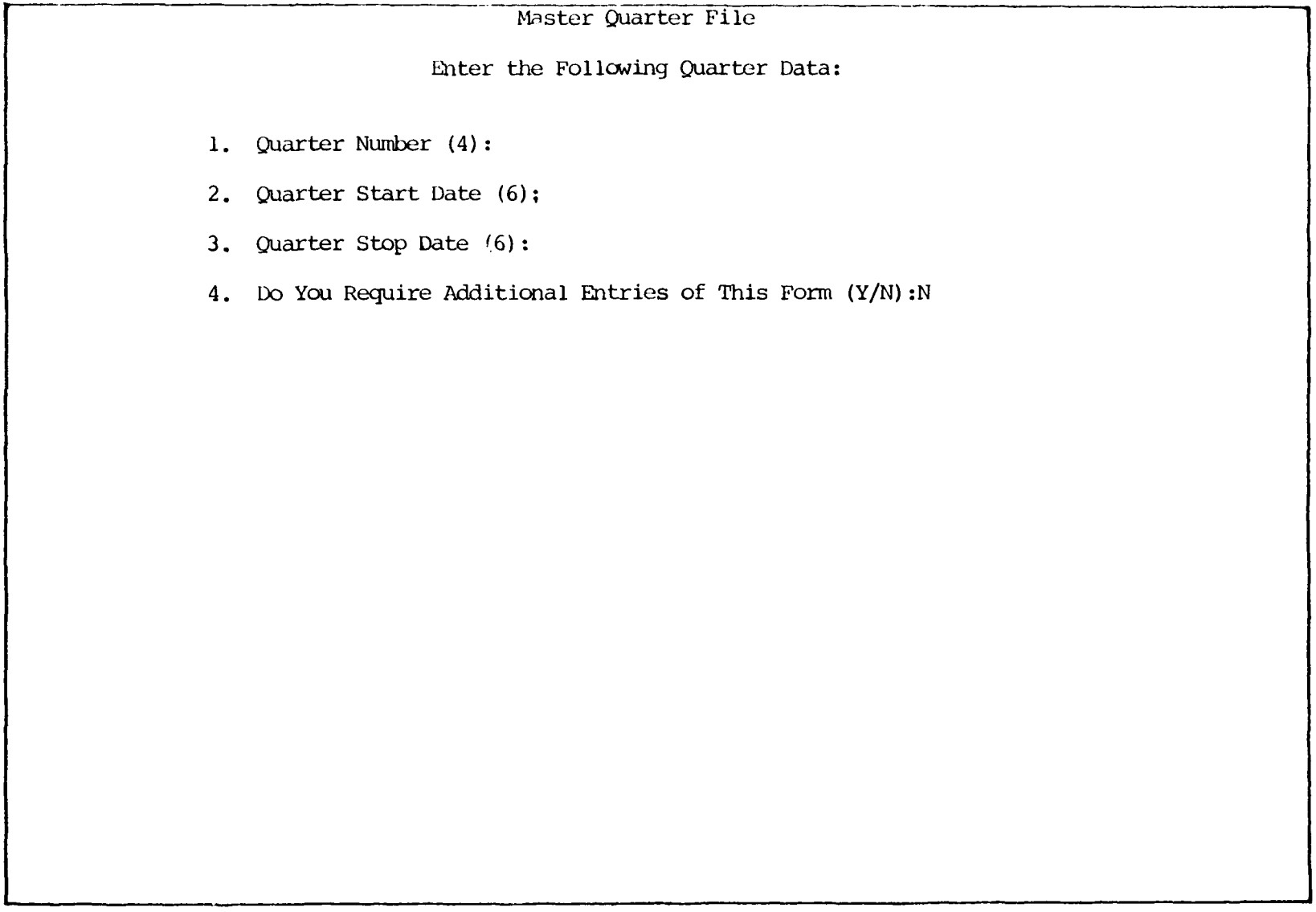


Figure I-29. Master Quarter Input Data

VARIABLE THESIS COMMITTEE MEMBER FILE

TCMF - (Links only)

VARIABLE FACULTY ADVISOR FILE

FADV - (Links only)

VARIABLE INSTRUCTOR STATISTICS FILE

VINS - (Links only)

VARIABLE THESIS ADVISOR FILE

TADV - (Links only)

VARIABLE PROFESSIONAL DEVELOPMENT FILE

VPDQ - (Links only)

VARIABLE SEQUENCE FILE

VMSS - Lists which courses belong in sequence

VARIABLE SECTION LEADER FILE

SECL - (Links only)

VARIABLE QUARTER FILE

VCQR - Coded 'QC' (Course data)
- Fill data
- Coded 'QS' (Student data - link)
- Grad data
- Coded 'FQ' (Faculty data - link)

VARIABLE REQUISITE FILE

VREQ - Coded 'CR' (Code is corequisite)
- Requisite course number
- Coded 'PR' (Code is prerequisite)
- Requisite course number

VARIABLE TEXT FILE

VTXT - Coded 'DS' (Code is description)
- Description data
- Coded 'OL' (Code is outline)
- Outline data

VARIABLE BOOK LINK FILE

VCBK - (Links only)

VARIABLE NUMBER ORDERED FILE

VNMO - Book title control field

VARIABLE CLASS SCHEDULE FILE

SCHD - Number of students in class
- Class finish time

VARIABLE CLASSROOM FILE

CLSR - Type(s) of equipment in room
- Code for type of room
- Code for security classification level of room

VARIABLE HONORS AND AWARDS FILE

NHAW - Coded 'HN' (Honor Area)
- Honors received
- Date honor received

- Coded 'AW' (Award Area)
- Awards received
- Date award received

VARIABLE FACULTY INTERESTS FILE

FINT - Area of interest

VARIABLE FACULTY PUBLICATIONS AND PRESENTATIONS

FCOM - Coded 'PB' (Publication data area)
- Title of publication
- Date of publication
- Name(s) of co-authors

- Coded 'PR' (Presentations data area)
- Presentation given to this organization (give organization)
- Presentation date

VARIABLE FACULTY TDY FILE

FTDY - Cost of TDY data in this file
- Destination
- Begin date
- End date

VARIABLE STUDENT COURSE FILE

CRSE - Course number
- Course name
- Course grade
- Quarter student took or will take course
- College attended
- Course waived (Y/N)?

VARIABLE THESIS TITLE FILE

THTL - Thesis title
- Thesis sponsor
- Thesis location
- Thesis classification

MASTER BUILDING/ROOM FILE

BLRM - Building and room number

MASTER ROOM CAPACITY FILE

CPTY - Capacity number

MASTER DAY SCHEDULING FILE

DAYS - Day of the week

MASTER SEQUENCE FILE

MSSF - Course sequence number
- Sequence name

MASTER DEGREE REQUIREMENTS FILE

MDEG - Number identifying type grad degree
- Name of type of degree

VARIABLE EDUCATION FILE

VEDU - Institution (University) attended
- Year degree awarded

VARIABLE FACULTY SOCIETY FILE

FSOC - Societies to which individual belongs

VARIABLE FACULTY DEPARTMENT AND COMMITTEE FILE

FCMT - Coded 'DP' (Committee membership)
- Name of committee
- Name of department
- Coded 'CM' (Additional committee memberships)
- Name of other committee
- Name of other department (if different)

MASTER THESIS NUMBER FILE

THES - Thesis cataloging number

MASTER SECTION NUMBER FILE

SECT - Section number (ex., GCS-84D)
- Section leader Social Security Number
- Graduation date
- Entry date
- Number of students in section

MASTER COURSE FILE

MCRS - Course number
- Course credit hours
- Course lecture hours data
- Course lab hour data
- Size limit data
- Title data
- Restricted (from grad requirements) course?

MASTER QUARTER FILE

MQTR - Quarter number
- Quarter start date
- Quarter stop date

MASTER BOOK FILE

MBKT - Book title name
- Book author name
- Book publisher name
- Number of books available
- Book price

MASTER ORDER FILE

MORD - Master order number
- Order number
- Due date
- Company order is with
- Company address
- Company phone number with area code

MASTER CLASS TIME FILE

TIME - Military clock time

- Office phone
- Last organization
- Last position title
- Expected AFIT departure date

MASTER DEPARTMENT FILE

DEPT - Department code
- Department name

MASTER STUDENT FILE

STDT - Student Social Security Number
- Name
- Mil/Civ Rank
- Has student already graduated/left AFIT?
- Military Service
- Aero rating
- Date of rank
- Date of commission
- Years of service
- Sex
- Box number
- Duty AFSC
- Primary AFSC
- Current address
- Emergency address
- Home phone
- Duty phone
- Education code
- Date of birth
- Place of birth
- Marital status
- Spouse first name
- Spouse date of birth
- Number of dependents
- Race
- Religion
- Losing command
- Last organization
- Last position title
- Duration at last duty assignment

MASTER NTIC NUMBER FILE

ADNR - Thesis code

Appendix J

Expanded AFITDB Data-set Items

The following are the Data-sets of the expanded AFIT Database, shown by data-set name and each data-item contained within it. Each data-item is listed in the order it appears in its data-set within the Data Base Generation (MIKEDB.DBG) document. The less important (to the academic administration) and less likely changed data (spouse name, date of birth, place of birth) is placed toward the end of the data-set. The order represents all master data-sets first followed by the variable data-sets, since during database generation, all referenced master data-sets must first be defined.

MASTER FACULTY FILE

FACT - Faculty Social Security Number
- Name
- Rank
- Military Service
- Date of Commission
- Date hired
- Salary
- Date of birth
- Sex
- Aero rating
- Duty AFSC
- Primary AFSC
- Date of rank
- Years of Service
- Current address
- Home phone
- Emergency address
- Marital status
- Spouse first name
- Spouse date of birth
- Number of dependents
- Race
- Religion
- Office room number

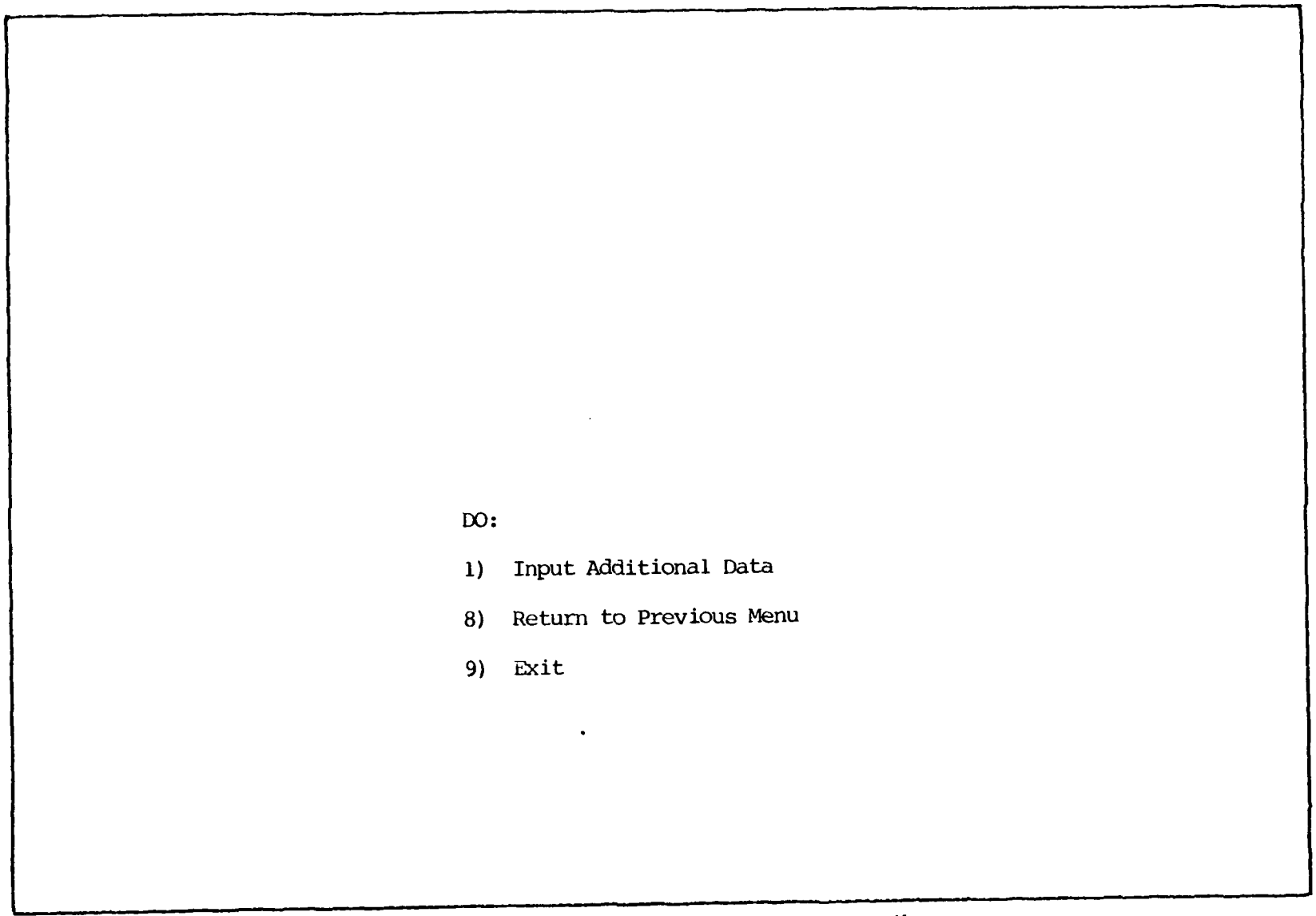


Figure I-36. Last Menu Screen Displayed From Library

School of Engineering Date (6):

Student Graduate Credit Record

Name (28): Rank (2): Class (8):

Min. Credit Hrs		Course (6)	Credit Hrs (2)	Grade (2)	Pts (3)	Initials
Thesis	(6)				.	
Math	(6)	1.			.	
		2.			.	
Sequence B		1.			.	
	(6)	2.			.	
		3.			.	
		4.			.	
		5.			.	
Sequence A		1.			.	
	(6)	2.			.	
		3.			.	
		4.			.	
		5.			.	
Other Approved Graduate Courses (6):						
		1.			.	
		2.			.	
		3.			.	

Figure I-35. Graduate Credit Record Form

Variable Sequence Listing

Enter the Course Number of Those Courses

In Course Sequence-
With Corresponding Course Sequence Number-

In the Area Below (5 Areas of Six Characters Each):

Figure I-34. Variable Sequence Listing

Degree Requirements

1. Enter the Degree Title (40):

2. Enter the Number to be Associated
   With the Degree Title Listed Above (2):

3. Do You Require Additional Entries of This Form (Y/N):N

Figure I-33. Degree Conversion Listing

Student Education Plan

Student's Name: Rank: AFSC: Section: Ed Code:

Thesis Title: Faculty Advisor:

Course Number:	Course Title:	Hrs	Grade	Pts
	Cum Hours:	Qtr Hrs:		
Course Number:	Course Title:	Hrs	Grade	Pts
	Cum Hours:	Qtr Hrs:		

Figure I-32. Student Education Plan Form (Form 29A)

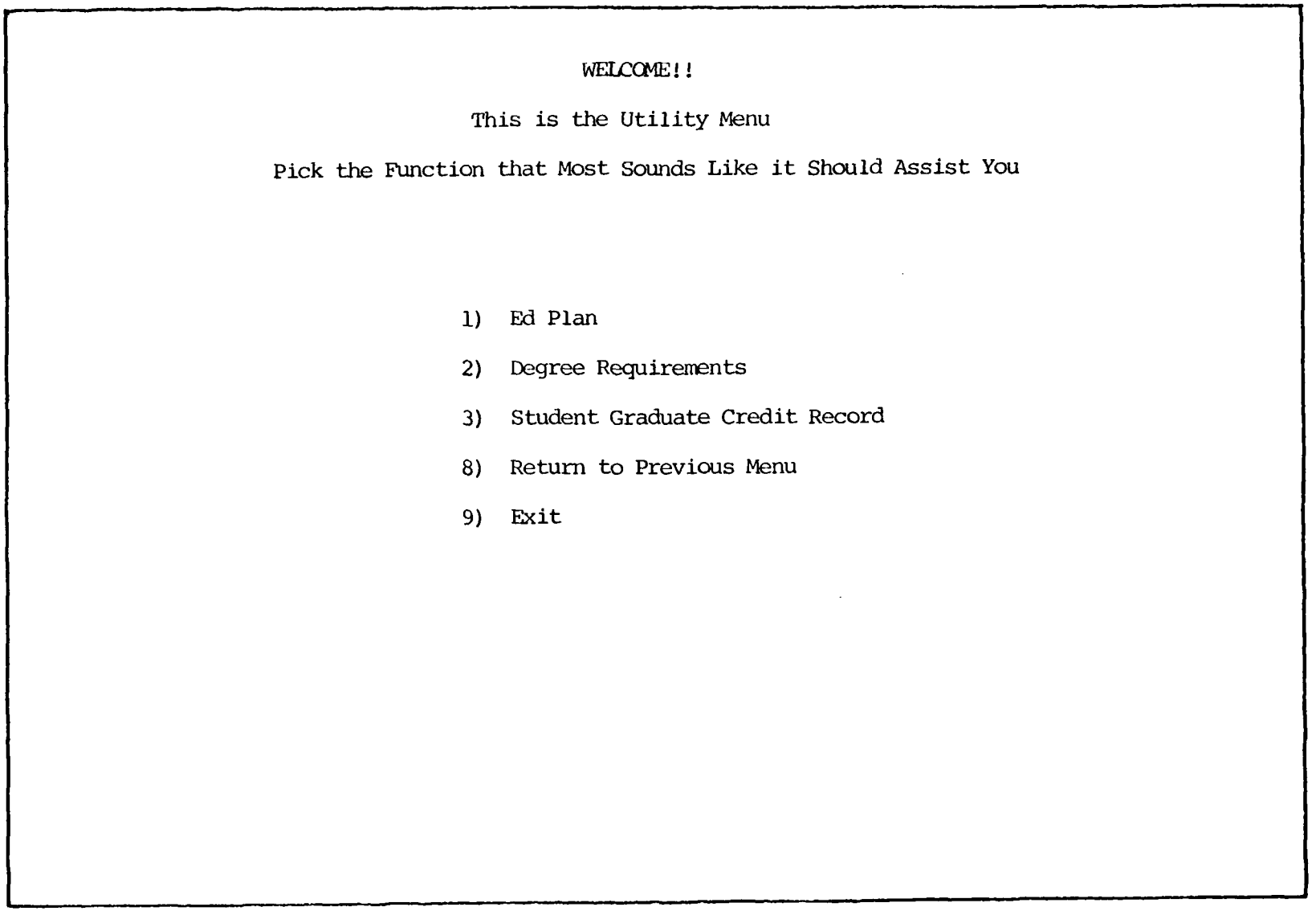


Figure I-31. Utility Directory

Course Sequences

1. Enter the Course Sequence Name (40):

2. Enter the Corresponding Course Sequence Number (3):

3. Do You Require Additional Entries of This Form (Y/N):N

Figure I-30. Course Sequence Data

Appendix K

AFITDB I/O Areas

The following lists I/O areas designations, number of copies for each I/O area, and those data-sets assigned to each I/O area. The number of copies for each I/O area were determined based on expected user usage as determined from the interview results listed in Appendix B and the prioritized listing of new database requirements in Appendix D. The configuration of the data-sets also contributed to the final determination.

Each data-set with the database is assigned a logical work-space area known as an I/O area. When a data-set or record is to be processed it is placed into the designated I/O area. The values of any data elements and/or items specified will be moved, one by one, from the corresponding record fields in the I/O buffer to a user-defined data storage area, like a buffer or array.

With only a single copy of an I/O area assigned to a file or files, buffer thrashing could occur with a corresponding impact on performance. Multiple copies, if present, will be allocated and used dynamically on a least recently used basis. A number of rules apply to I/O areas:

1. No duplicate (I/O area) names are allowed.

2. Each data-set may have one and only one I/O area assigned to it when the data set is defined.

3. A master data-set must not share an I/O area with a variable-entry data-set.

4. An I/O area assigned to one and only one data-set must be listed and is regarded as a private I/O area.

5. Each occurrence of this statement will reserve an area whether or not it is ever referenced in the data-set definition.

6. There is a limit of 15 areas that may be defined.

7. A maximum of 32 copies of an I/O area is possible with a default of 1.

(6:4-6)

Each I/O area is listed under the left column with the breakout of the data-sets to their right. Multiple copies are designated by the equals immediately to the right of the listed I/O area (ex., MAS3= 2). They are subjectively grouped according to anticipated use and expected relatedness in future application programs. The concept is that heavily accessed and large data-sets have their own I/O area as well as unrelated and smaller data-sets. This breakout is expected to assist in reducing extensive I/O handling.

MASTER I/O AREAS	DATA-SET
MAS1	MBKT; MORD
MAS2	DEPT; ADNR
MAS3=2	MCRS
MAS4=2	STDT
MAS5=2	FACT
MAS6=2	THES; SECT
MAS7	MQTR
MAS8=2	MSSF; MDEG
MAST	TIME; BLRM: CPTY; DAYS

VARIABLE I/O
AREAS

VARX	VCQR; VREQ; VTXT; VCBK; VNMO
VAR1	VEDU; FSOC; FCMT; VHAW; FINT; FCOM; FTDY
VAR3=2	VPDQ
VAR4=2	SECL; TCMF; FADV; TADV
VAR5=2	CRSE; THTL
VAR6=2	VCQR; VMSS
VART	SCHD; CLSR

Appendix L

AFITDB Data-set Drive Designations

Each data-set requires a specific disk area be defined for use during database interaction. The following specifies how the drive number and data-sets interact and lists the separate drive numbers for each data-set. Failure to specify separate drive identifiers causes a logical error within the initiation of the data-sets.

The name of each data-set is first listed with the corresponding drive data following. The format is 'Drive = n,n,xxn' with:

n — specifies the logical unit number to be associated with this section.

n — specifies the number of physical sectors to allocate with this section.

xxn — specifies the device and unit number to allocate this section. In this case 'DU1' indicates the drive which contains the database and DBMS.

The logical unit number must be unique within the data base. Multiple drive statements may be specified if the data-set requires more than one section. The maximum number of drive statements per data-set is 32. The number of sectors used within any section is always a multiple of the sectors per block. The value of sectors per block is calculated internally depending upon the values specified for {logical-record-length} and {logical-records-per-block.} If these two statements are not included {DBGEN} (database generation routine) will use a default value of 1 for sectors per block. The number of sectors actually allocated may differ from the number specified. {DBGEN} will only allocate as much space as defined by the {TOTAL-Logical-Records} statements. The value for sectors may not exceed 32,767 (one less than the maximum DBMOD size) in any one section

Therefore, data-set 'TIME' is associated with logical unit number 01, can have at most 5000 sectors allocated and will reside on device DU1 (DUA1 - a Winchestor drive). The number of sectors was selected to be 5000 to insure adequate record space for data-set expansion. As stated above, TOTAL only allocates sufficient space to accommodate the anticipated number of stored records ({TOTAL-Logical-Records}). The listing is by logical number with no data-set name listed for unassigned logical units (6:4-20).

```
DATA-SET-NAME=TIME          DRIVE=01,5000,DU1

DATA-SET-NAME=BLRM          DRIVE=02,5000,DU1

DATA-SET-NAME=CPTY          DRIVE=03,5000,DU1
```

DATA-SET-NAME=DAYS	DRIVE=04,5000,DU1
DATA-SET-NAME=SCHD	DRIVE=05,5000,DU1
DATA-SET-NAME=CLSR	DRIVE=06,5000,DU1
DATA-SET-NAME=	DRIVE=07,5000,DU1
DATA-SET-NAME=	DRIVE=08,5000,DU1
DATA-SET-NAME=	DRIVE=09,5000,DU1
DATA-SET-NAME=	DRIVE=10,5000,DU1
DATA-SET-NAME=	DRIVE=11,5000,DU1
DATA-SET-NAME=MCRS	DRIVE=12,5000,DU1
DATA-SET-NAME=MQTR	DRIVE=13,5000,DU1
DATA-SET-NAME=MBKT	DRIVE=14,5000,DU1
DATA-SET-NAME=MORD	DRIVE=15,5000,DU1
DATA-SET-NAME=VCQR	DRIVE=16,5000,DU1
DATA-SET-NAME=VREQ	DRIVE=17,5000,DU1
DATA-SET-NAME=VTXT	DRIVE=18,5000,DU1
DATA-SET-NAME=VCBK	DRIVE=19,5000,DU1
DATA-SET-NAME=VNMO	DRIVE=20,5000,DU1
DATA-SET-NAME=ADNR	DRIVE=21,5000,DU1
DATA-SET-NAME=THES	DRIVE=22,5000,DU1
DATA-SET-NAME=SECT	DRIVE=23,5000,DU1
DATA-SET-NAME=THTL	DRIVE=24,5000,DU1
DATA-SET-NAME=SECL	DRIVE=25,5000,DU1
DATA-SET-NAME=STDT	DRIVE=26,5000,DU1
DATA-SET-NAME=CRSE	DRIVE=27,5000,DU1
DATA-SET-NAME=AWAR	DRIVE=28,5000,DU1
DATA-SET-NAME=DGYR	DRIVE=29,5000,DU1

DATA-SET-NAME=FACT	DRIVE=30,5000,DU1
DATA-SET-NAME=DEPT	DRIVE=31,5000,DU1
DATA-SET-NAME=FEDU	DRIVE=32,5000,DU1
DATA-SET-NAME=FSOC	DRIVE=33,5000,DU1
DATA-SET-NAME=FCMT	DRIVE=34,5000,DU1
DATA-SET-NAME=	DRIVE=35,5000,DU1
DATA-SET-NAME=FHAW	DRIVE=36,5000,DU1
DATA-SET-NAME=FINT	DRIVE=37,5000,DU1
DATA-SET-NAME=FCOM	DRIVE=38,5000,DU1
DATA-SET-NAME=FTDY	DRIVE=39,5000,DU1
DATA-SET-NAME=VSEQ	DRIVE=40,5000,DU1
DATA-SET-NAME=MSEQ	DRIVE=41,5000,DU1
DATA-SET-NAME=MSSF	DRIVE=42,5000,DU1
DATA-SET-NAME=MDEG	DRIVE=43,5000,DU1
DATA-SET-NAME=TCMF	DRIVE=44,5000,DU1
DATA-SET-NAME=FADV	DRIVE=45,5000,DU1
DATA-SET-NAME=VINS	DRIVE=46,5000,DU1
DATA-SET-NAME=TADV	DRIVE=47,5000,DU1
DATA-SET-NAME=VPDQ	DRIVE=48,5000,DU1
DATA-SET-NAME=VMSS	DRIVE=49,5000,DU1

Appendix M

Functional Requirements and System Design Documentation

The following represents the functional requirements determined initially during the analysis stage of the project. The four major headings, marked by the integer numbered items 1.0 through 4.0, show the processes determined to be of prime importance during the testing of the Driver and follow-on application programs.

1.0 Establish a user interface
- 1.1 Log on database properly
- 1.2 Allow user to select type of database operation
- 1.3 Allow user to inspect selection
- 1.4 Allow user to specify either faculty or student database use
- 1.5 Allow user to specify types of utility to be conducted
- 1.6 Allow user to add/change/modify data

2.0 Set up selected data manipulation routine
- 2.1 Create or initialize data files
- 2.2 Map selected utility to proper application program
- 2.3 Minimize the amount of variables, links, data sets looked at during data manipulation
- 2.4 Return error status to user

3.0 Close out the data session
- 3.1 Allow user to gracefully terminate session
- 3.2 Deallocate resources
- 3.3 Return error status to user to forestall unnecessary delays in database usage
- 3.4 Log off database properly

4.0 Present data to user
- 4.1 Create output forms
- 4.2 Print output forms by user selected device
- 4.3 Display output data on terminal

(22)

Throughout the coding and implementation of the Driver program, test procedures were conducted using the individuals who had been interview subjects at the beginning of this project. When warranted, additional changes to design were made after such discussions (such as allowing the

user to exit the Driver program prior to the 'Firstl' screen, before the 'Signon' procedure).

Such test procedures were arrainged into test modules in order to process an orderly code test procedure within the project. The functional requirements specified in the first portion of this appendix are the basis of the test plan presented herein.

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 1.1 - Log on database properly.

TEST CASE(S):

1. Attempt log-on without activating TOTALINIT.
2. Attempt log-on without TOTPRM.
3. Attempt log-on with files locked.
4. Attempt log-on without Schema defined.

EXPECTED RESPONSE:

1. Program will run without database interaction.
2. 'Database not found' will be returned to program.
3. Trying to load locked files will return lock error.
4. Log-on will cause catestrophic failure and complete unload of attempted operation.

RESULTS:

CASE 1. - PASS:_____	FAIL:_____	DATE:_____
CASE 2. - PASS:_____	FAIL:_____	DATE:_____
CASE 3. - PASS:_____	FAIL:_____	DATE:_____
CASE 4. - PASS:_____	FAIL:_____	DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 1.2 - Allow user to select type of database operation.

TEST CASE(S):

1. No selection entered.
2. Wrong type data entered.
3. 'Exit' selected.

EXPECTED RESPONSE:

1. Unless 'Exit' selected, user unable to advance until proper response made.
2. Data not accepted for input, terminal signals this to user in form of bell.
3. If choice is 'Exit', shut down database and log off.

RESULTS:

CASE 1. - PASS:_____	FAIL:_____	DATE:_____
CASE 2. - PASS:_____	FAIL:_____	DATE:_____
CASE 3. - PASS:_____	FAIL:_____	DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 1.3 - Allow user to inspect selection.

TEST CASE(S):

1. Enter selection into field.
2. For multi-field form, allow back-up to enter or change field data.

EXPECTED RESPONSE:

1. Control of program is not returned until terminator is initiated by user.
2. Use of backspace key allows user to return to previously defined fields.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 1.4 - Allow user to specify either faculty or student database use.

TEST CASE(S):

1. Enter allowed digit.
2. Enter more than one digit.
3. Enter other than digit.
4. Enter other than allowed digit.

EXPECTED RESPONSE:

1. Entry accepted and waits for terminator.
2. Terminal displays ill-will by 'beeping' and subsequent digits not accepted.
3. Terminal 'beeps' and displays messsage in line 25 signifying user to enter digit.
4. Message displayed that entry is invalid and control returned to beginning of selection process.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____
CASE 3. - PASS:_____ FAIL:_____ DATE:_____
CASE 4. - PASS:_____ FAIL:_____ DATE:_____

TESTED:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 1.5 - Allow user to specify types of utility to be conducted.

TEST CASE(S):

1. Enter allowed digit.
2. Enter more than one digit.
3. Enter other than digit.
4. Enter other than allowed digit.

EXPECTED RESPONSE:

1. Entry accepted and waits for terminator.
2. Alert user with aural signal and subsequent digits not accepted.
3. Alert user with aural signal and display message in line 25 signifying user to enter digit.
4. Message displayed that entry is invalid and control returned to beginning of selection process.

RESULTS:

CASE 1. - PASS:_____	FAIL:_____	DATE:_____
CASE 2. - PASS:_____	FAIL:_____	DATE:_____
CASE 3. - PASS:_____	FAIL:_____	DATE:_____
CASE 4. - PASS:_____	FAIL:_____	DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 1.6 - Allow user to add/modify data.

TEST CASE(S):

1. If entry data see Test Case 1.2 - 1.5.
2. Add individual data to database.
3. Modify data in database.
4. Change SSN of individual after calling for data.
5. Enter data not intended into form field.
6. Enter more than expected data into field.
7. Enter less than specified data.
8. Fail to fill a 'Must-fill' field.

EXPECTED RESPONSE:

1. Same as in Test Case 1.2 - 1.5.
2. Enter data into form fields.
3. Strike over data already contained in field; abide by field characteristics as to field input.
4. Return status of illegal procedure and reject operation.
5. Alert user with aural signal and message to line 25 on screen.
6. Reject data of more than specified field length and sound aural signal.
7. Accept data unless violation of No. 8.
8. Prior to form terminator accepted, place cursor into 'Must-fill' field and sound aural signal.

RESULTS:

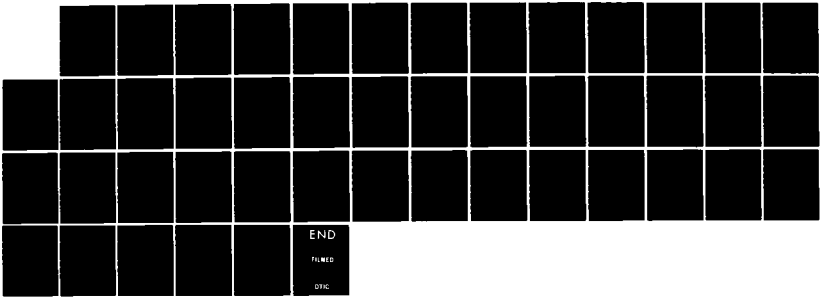
CASE 1. -	PASS:_____	FAIL:_____	DATE:_____
CASE 2. -	PASS:_____	FAIL:_____	DATE:_____
CASE 3. -	PASS:_____	FAIL:_____	DATE:_____
CASE 4. -	PASS:_____	FAIL:_____	DATE:_____
CASE 5. -	PASS:_____	FAIL:_____	DATE:_____
CASE 6. -	PASS:_____	FAIL:_____	DATE:_____
CASE 7. -	PASS:_____	FAIL:_____	DATE:_____
CASE 8. -	PASS:_____	FAIL:_____	DATE:_____

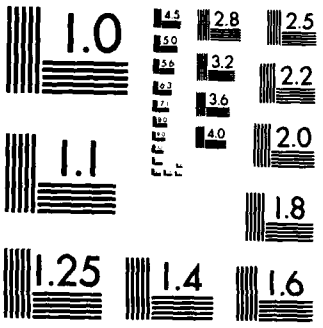
TESTED BY:_____

REMARKS:

AD-A153 043 COMPLETE DEVELOPMENT AND IMPLEMENT AFIT/ENG DATABASE MANAGEMENT SYSTEM VOLUME 1(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. M E PANGMAN 3/3

UNCLASSIFIED DEC 84 AFIT/GCS/ENG/84D-20 F/G 5/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 2.1 - Create or initialize data files.

TEST CASE(S):

1. Select operation for individual not entered into database.
2. Select operation for individual entered into database.

EXPECTED RESPONSE:

1. Individual not found; displays form to allow user to input data for inclusion into database.
2. Individual found; displays individual form data contained within database.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 2.2 - Map selected utility to proper application program.

TEST CASE(S):

1. Select one of offered choices.
2. Select choice not offered.
3. Enter non-sensical choice.
4. Enter no choice.

EXPECTED RESPONSE:

1. Proper utility corresponding with selection is begun.
2. Sound aural signal and display message in line 25 of screen.
3. Sound aural signal and display message in line 25 of screen.
4. Sound aural signal and display message in line 25 of screen.

RESULTS:

CASE 1. -	PASS:_____	FAIL:_____	DATE:_____
CASE 2. -	PASS:_____	FAIL:_____	DATE:_____
CASE 3. -	PASS:_____	FAIL:_____	DATE:_____
CASE 4. -	PASS:_____	FAIL:_____	DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 2.3 - Minimize the amount of variables, links, data sets looked at during data manipulation.

TEST CASE(S):

1. Log on database for single-user operation.
2. Log on database for multi-user operation.

EXPECTED RESPONSE:

1. Able to manipulate database with single user; with multiple users able to manipulate database data-sets of those data-sets not held by another user.
2. Able to manipulate like operations simultaneously.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 2.4 - Return error status to user.

TEST CASES(S):

1. Call TOTAL and display error code.
2. Display error code for those codes specified in STDTRESULTS.
3. Display error code for error not listed in STDTRESULTS.
4. Check for returned error after each type DATBAS call.

EXPECTED RESPONSE:

1. Four character error code displayed.
2. After specific module utilizing DATBAS call, error code stored in STDTRESULTS, which matches returned code, is displayed.
3. After specific module utilizing DATBAS call, error code not stored in STDTRESULTS is displayed.
4. Error status or '****' returned after each DATBAS call.

RESULTS:

CASE 1: - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____
CASE 3. - PASS:_____ FAIL:_____ DATE:_____
CASE 4. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 3.1 - Allow user to gracefully terminate session

TEST CASE(S):

1. Terminate an active database interaction session.
2. Attempt to terminate session using non-standard means.

EXPECTED RESPONSE:

1. Session terminated, all database transactions completed and database updated.
2. Session terminated, files in use remain locked and database nct closed.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 3.2 - Deallocate resources.

TEST CASE(S):

1. Terminate TOTAL database session and return to inactive status.

EXPECTED RESPONSE:

1. TOTAL updates database, unlocks used data-sets, and properly logs off the user(s).

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 3.3 - Return error status to user to forestall unnecessary delays in database usage.

TEST CASE(S):

1. Sign-on to database with database specified.
2. Sign-on to database without database specified.

EXPECTED RESPONSE:

1. Normal operation of database usage.
2. Normal operation of application program up to Sign-on and no database operation.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 3.4 - Log off database properly.

TEST CASE(S):

1. Follow menu process.
2. Exit using non-standard procedures.

EXPECTED RESPONSE:

1. Normal Sign-off procedure.
2. Returns use of terminal to user, but fails to update database, unlock user data-sets, or Sign-off from TOTAL.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 4.1 - Create output forms.

TEST CASE(S):

1. No report requested.
2. A report requested.

EXPECTED RESPONSE:

1. No report generated.
2. A report generated.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 4.2 - Print output forms by user selected device.

TEST CASE(S):

1. No device selected.
2. Printer device selected.
3. Terminal selected.

EXPECTED RESPONSE:

1. Output displayed to terminal.
2. One copy printed to selected printer device.
3. Output displayed to terminal.

RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____
CASE 3. - PASS:_____ FAIL:_____ DATE:_____

TESTED BY:_____

REMARKS:

SYSTEM DESIGN DOCUMENTATION

TEST PLAN

REQUIREMENT: 4.3 - Display output data on terminal.

TEST CASE(S):

1. No terminal display requested.
2. Terminal display requested.

EXPECTED RESPONSE:

1. No terminal display presented.
2. Output from database utility displayed to terminal.

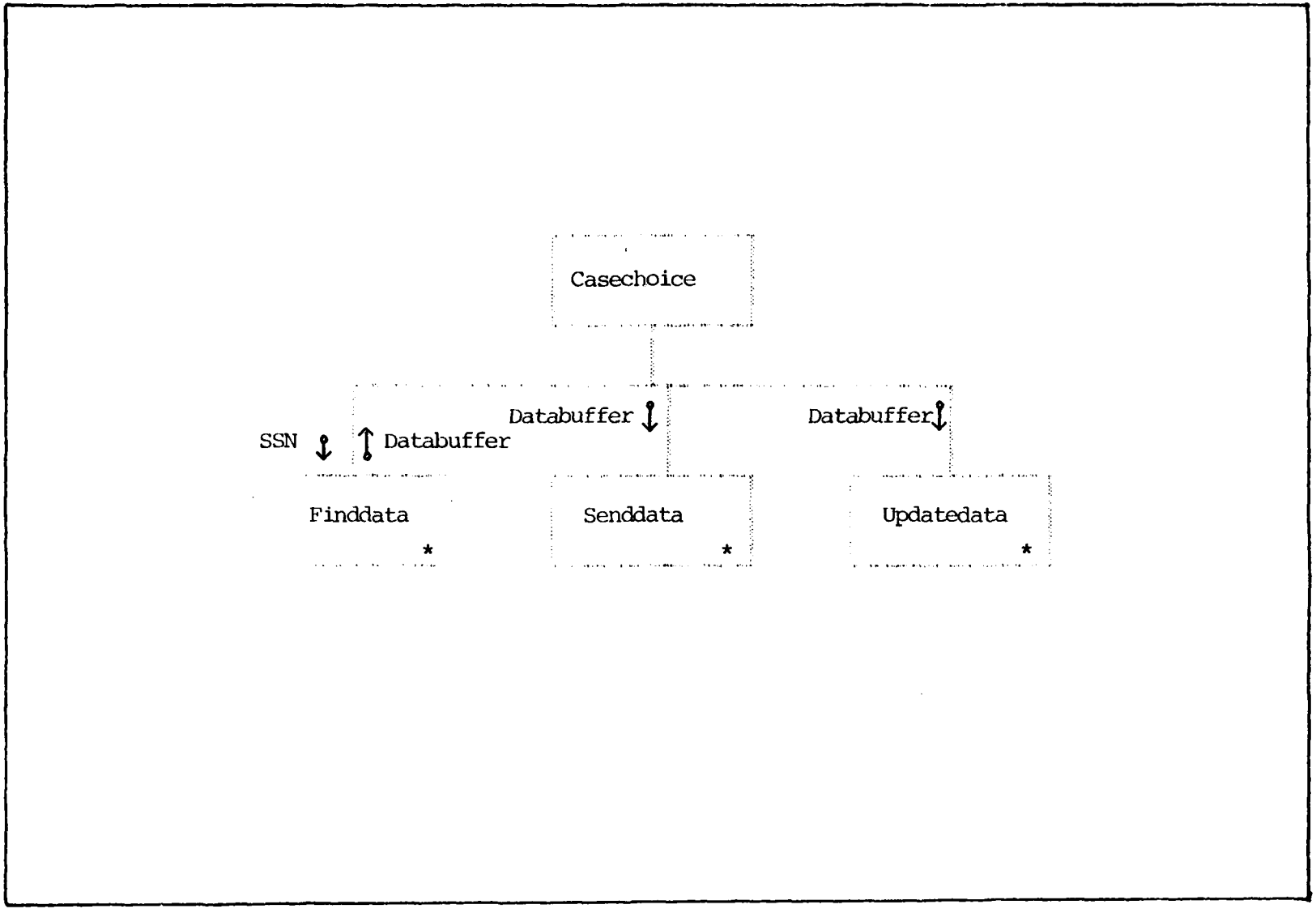
RESULTS:

CASE 1. - PASS:_____ FAIL:_____ DATE:_____
CASE 2. - PASS:_____ FAIL:_____ DATE:_____

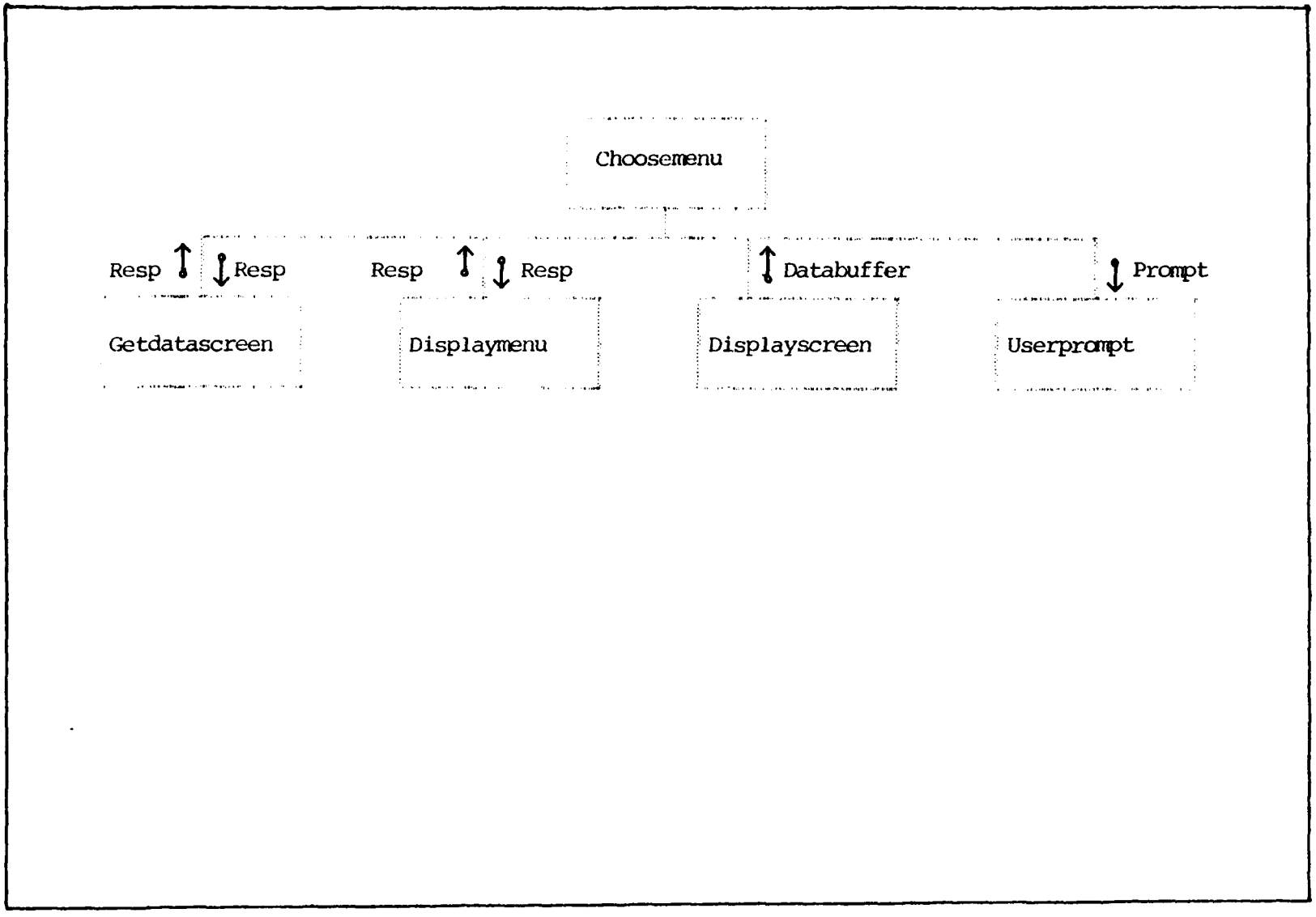
TESTED BY:_____

REMARKS:

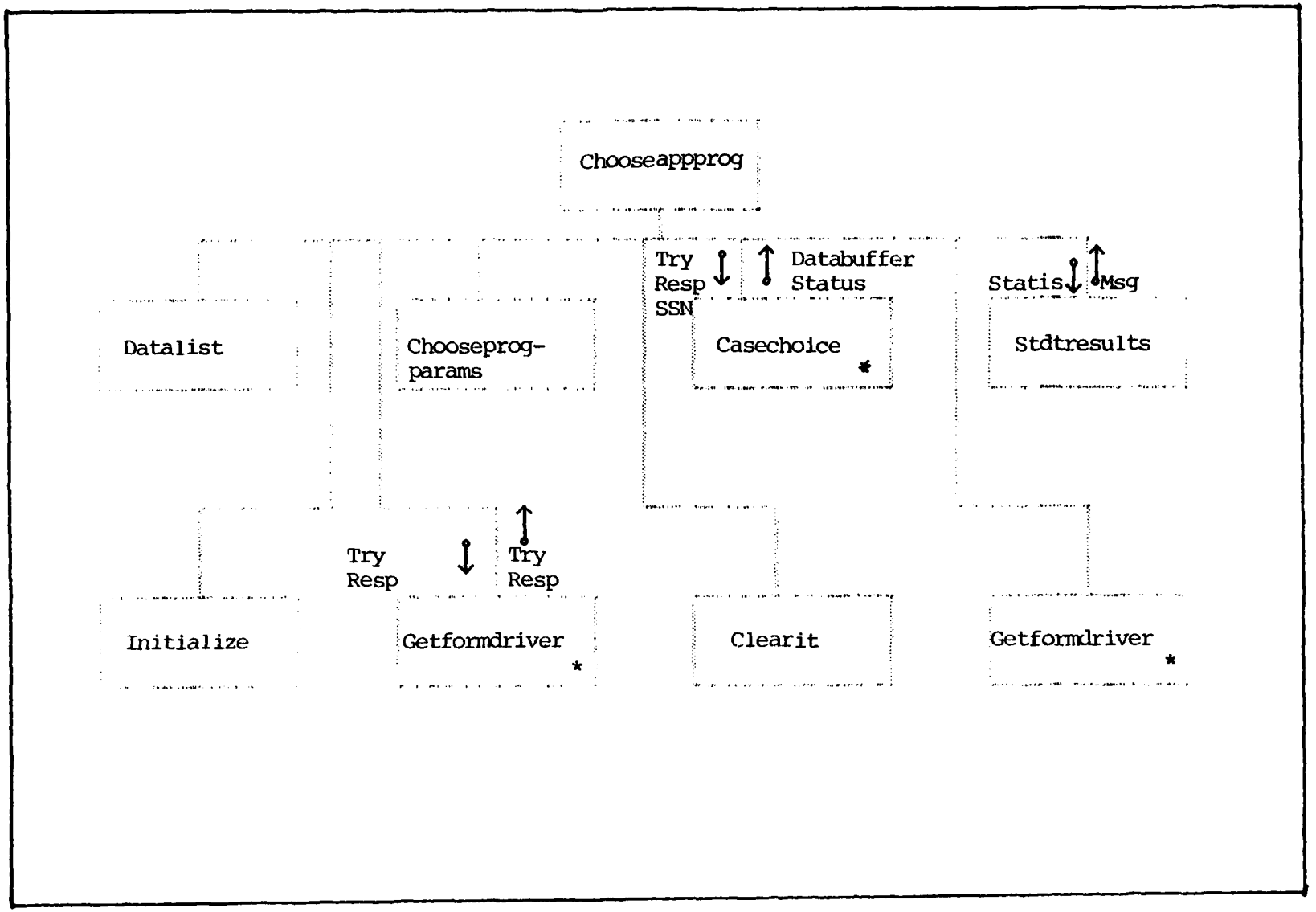
The following begins the System Design documentation consisting of Data Flow Diagrams and Structure Chart information. Each graphic section contains a diagram index and the graphic representations. The documents specify the order of the anticipated database drive package including expansion capability.



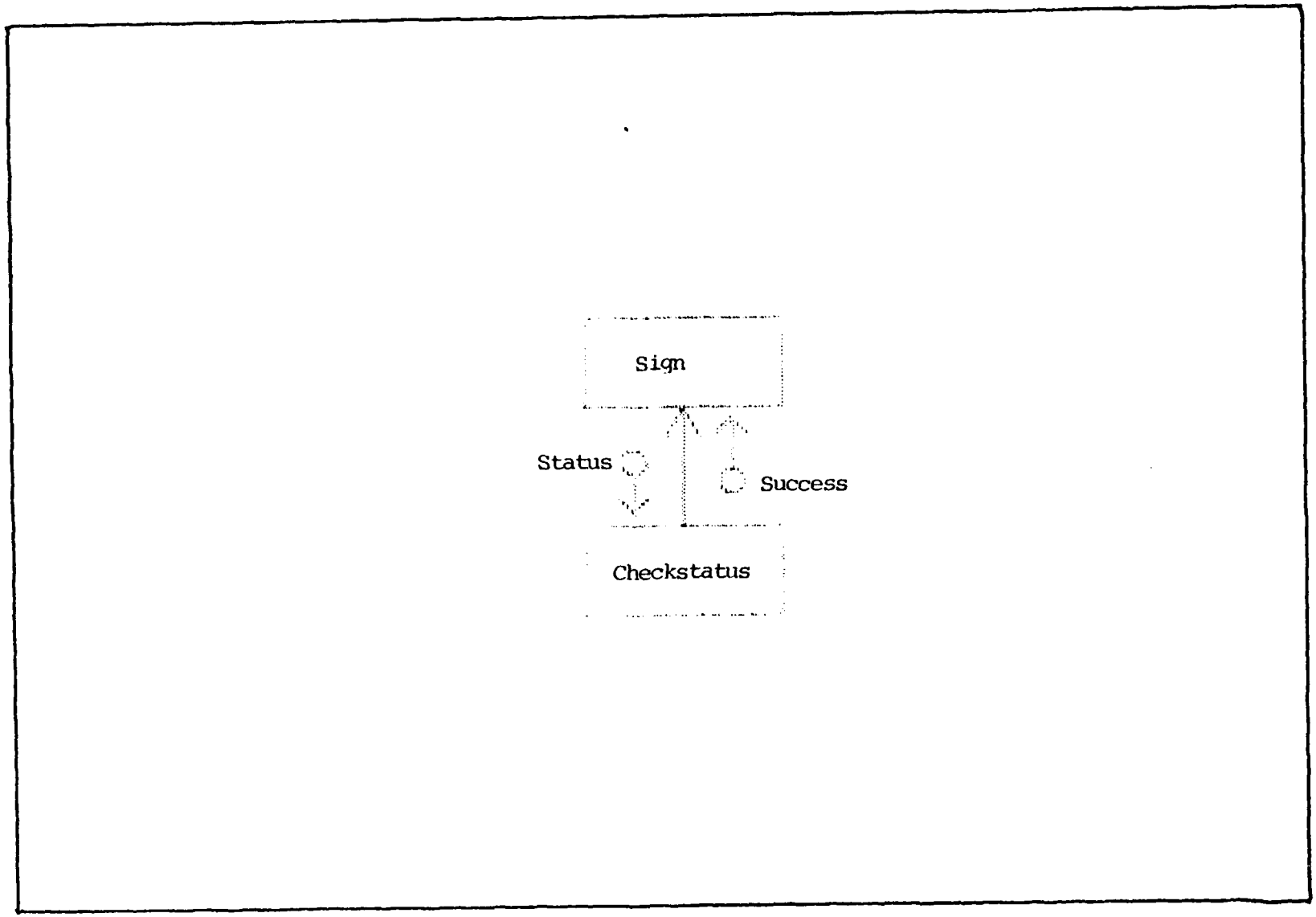
Structure Charts -- User Interface



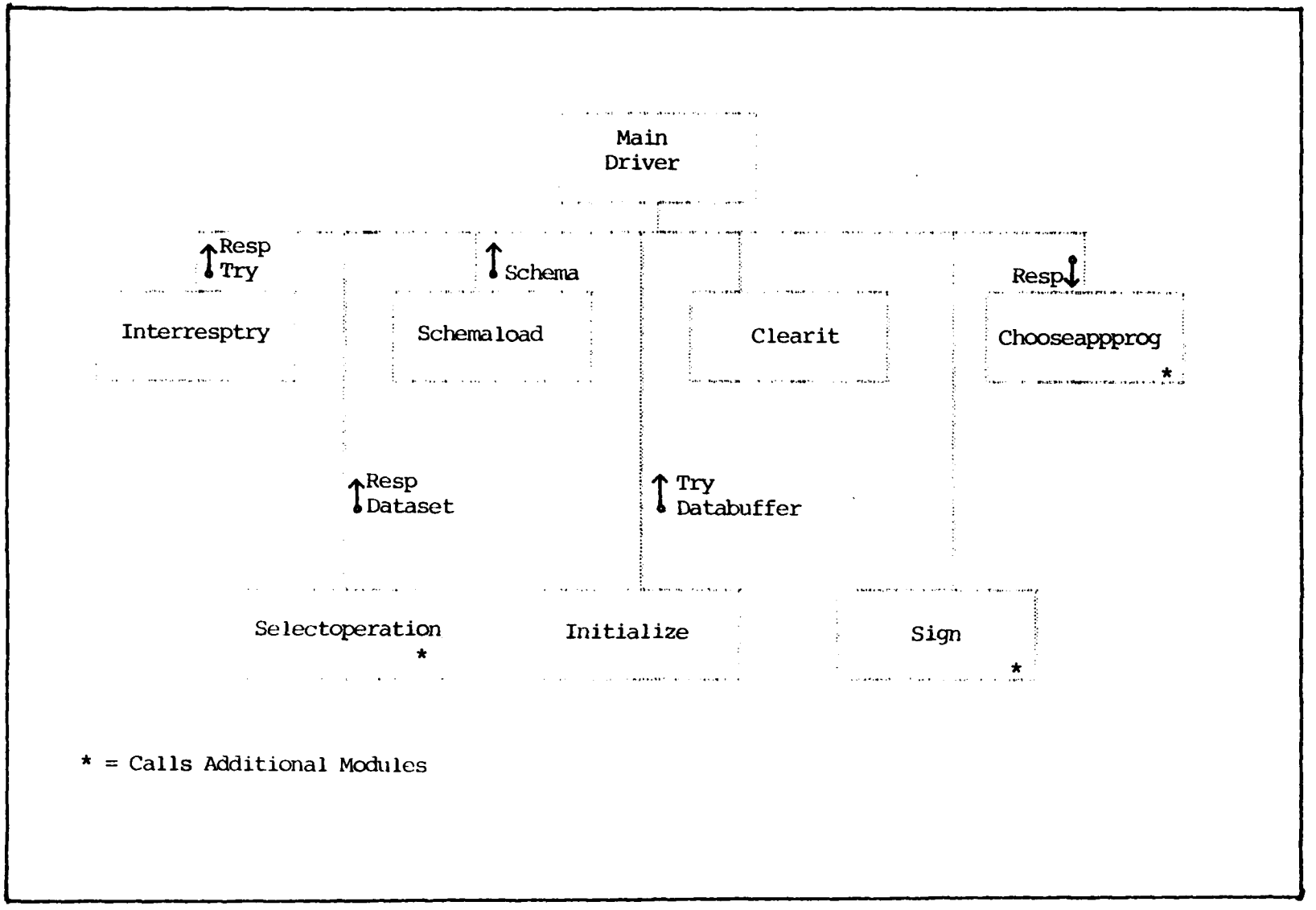
Structure Charts -- User Interface



Structure Charts -- User Interface



Structure Charts -- User Interface



Structure Charts -- User Interface

Structure Diagram Index

This portion contains the Structure Diagrams for the user interface of the expanded AFIT/ENG database. As with the Data Flow Diagrams (above), the documents specify the order of the anticipated database driver package including its expansion capability.

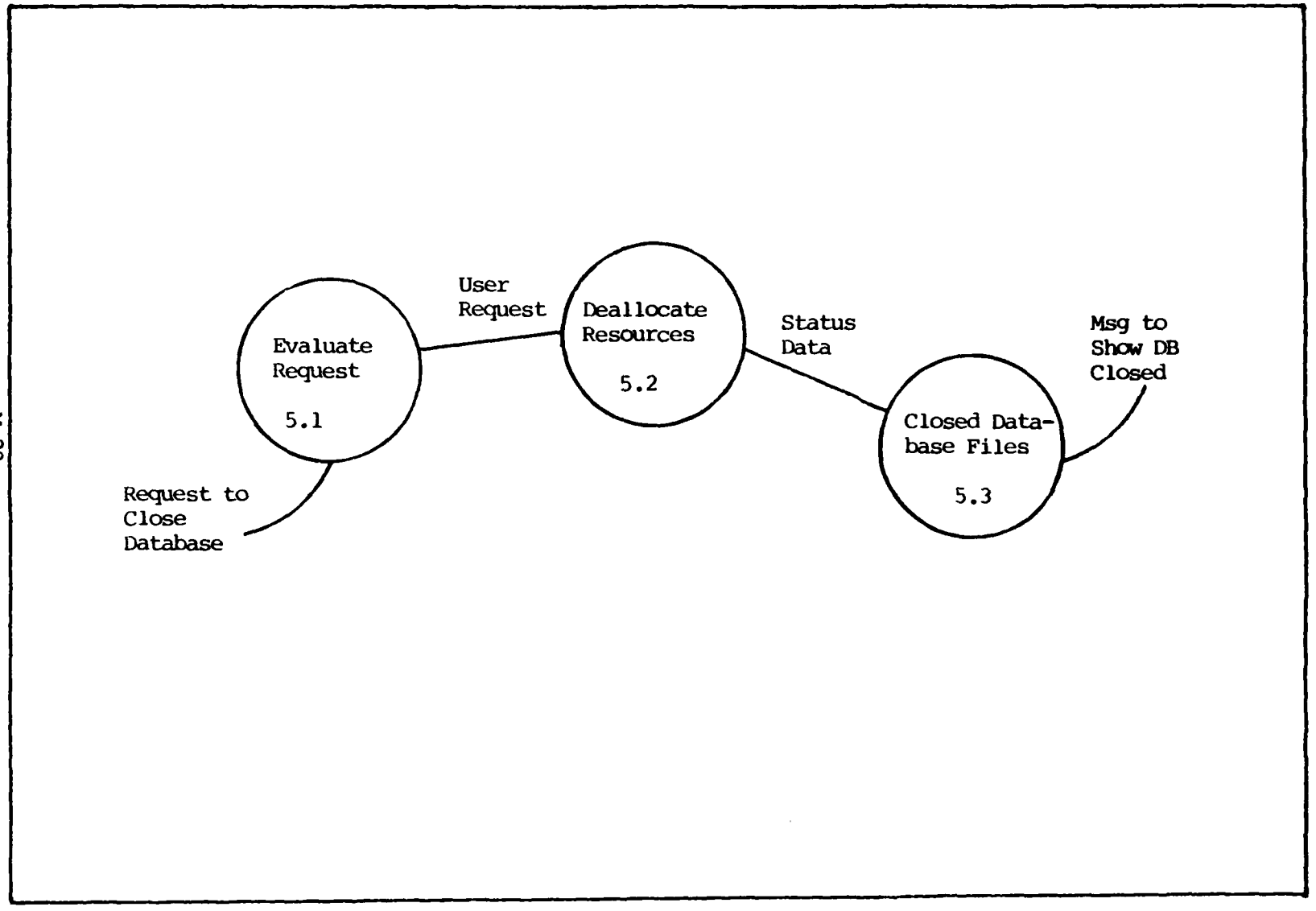


Figure M-7. Close Database (Level 5.1)

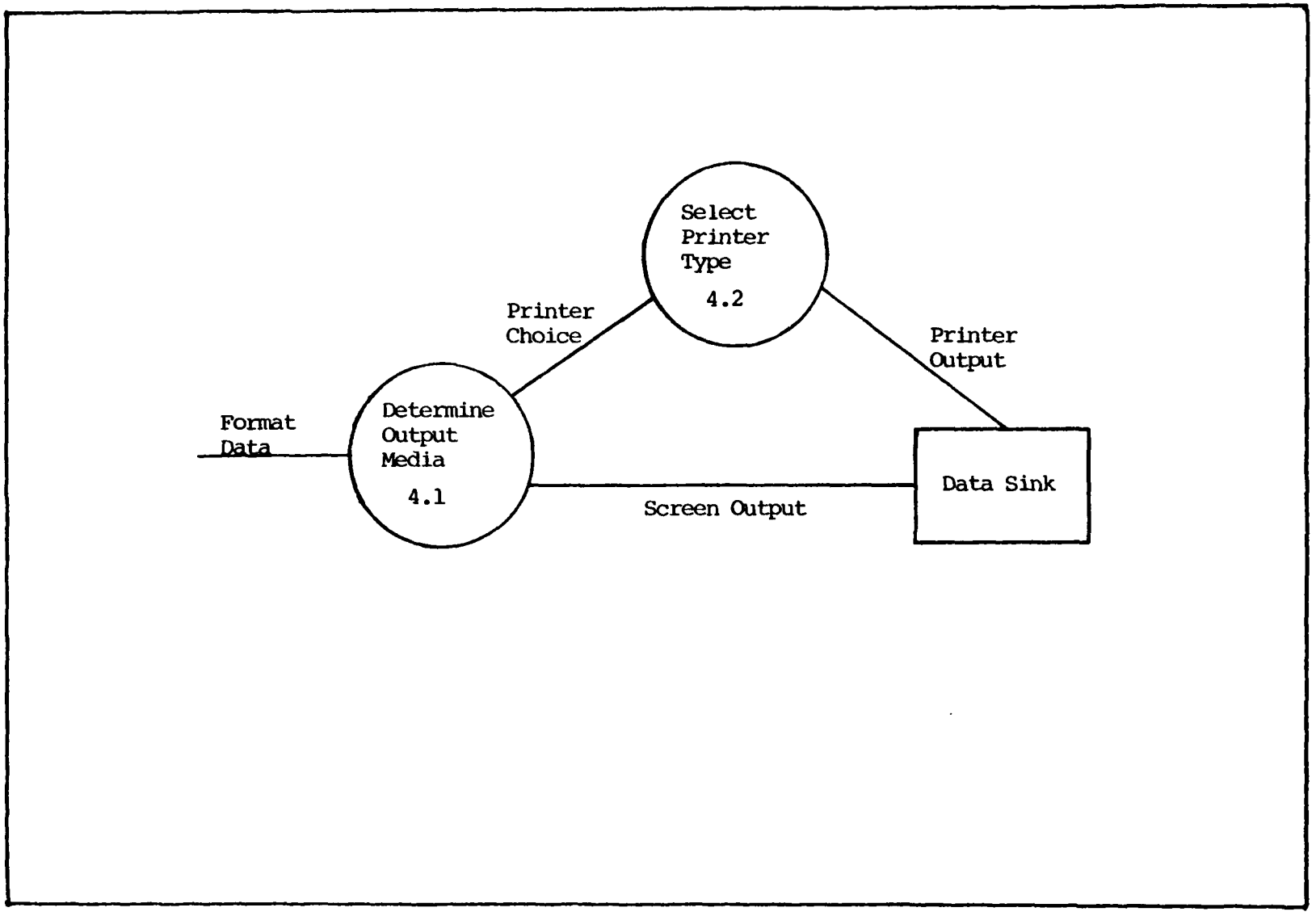


Figure M-6. Provide Output to User (Level 4.1)

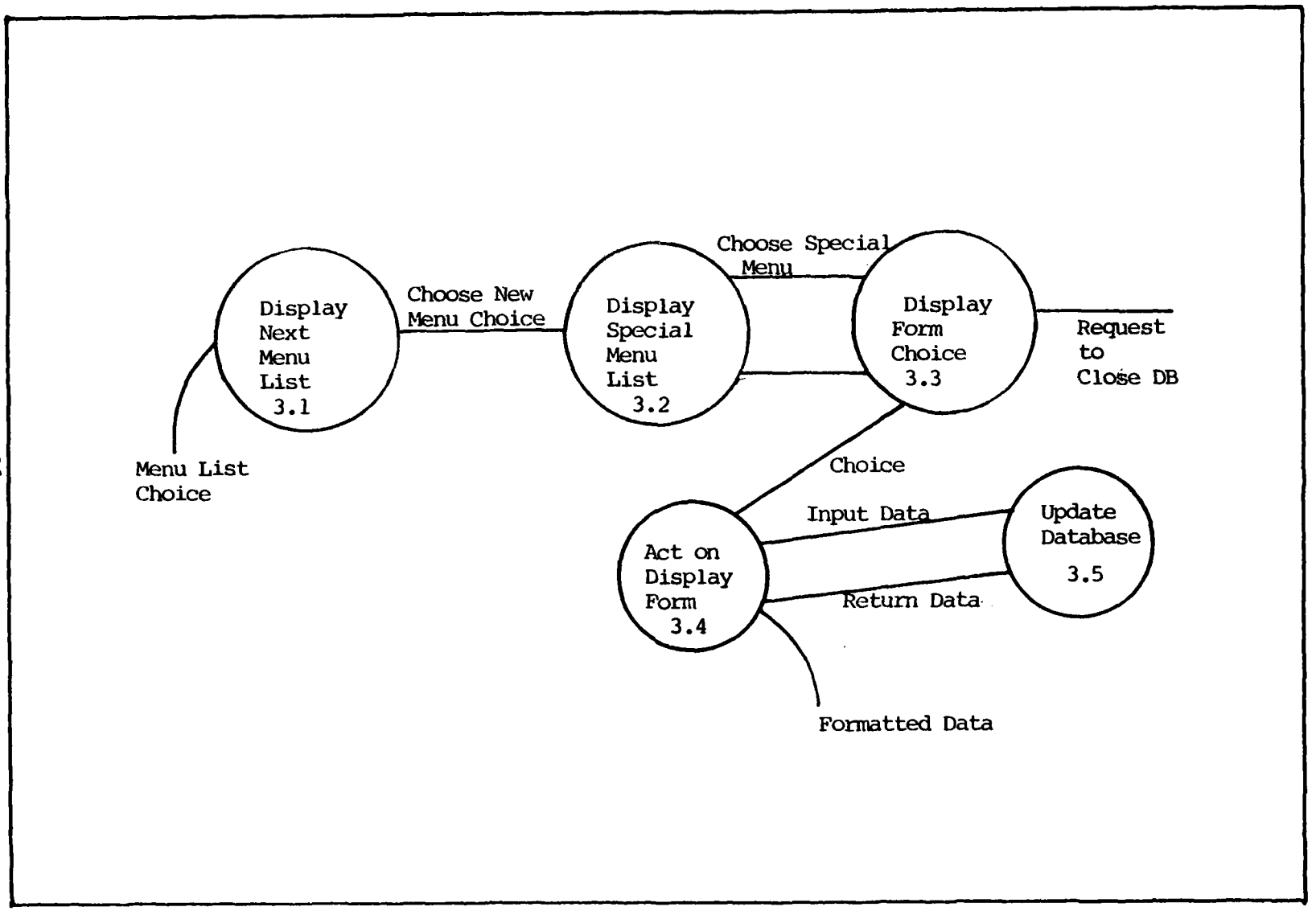


Figure M-5. Service Request (Level 3.1)

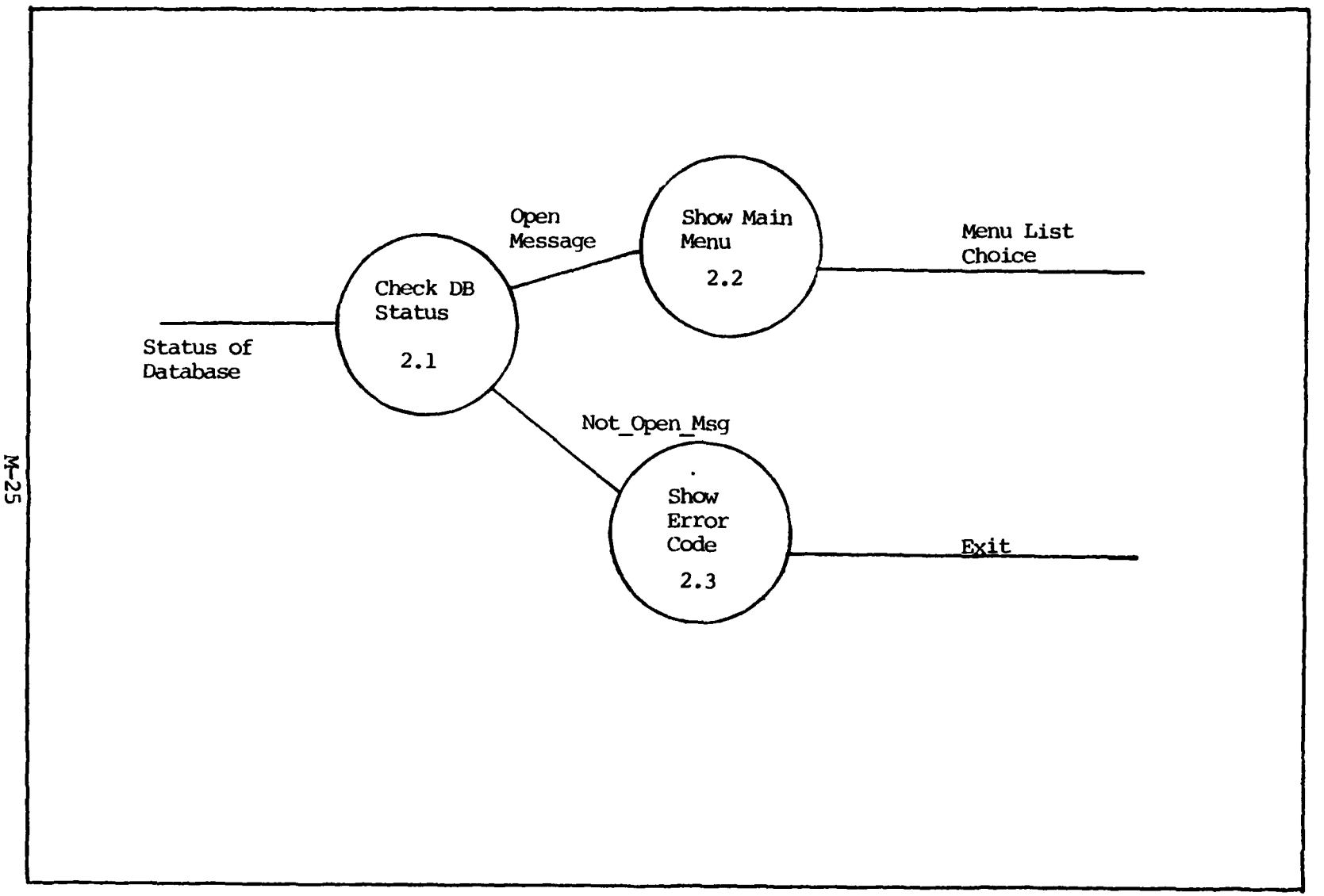


Figure M-4. Determine Request (Level 2.1)

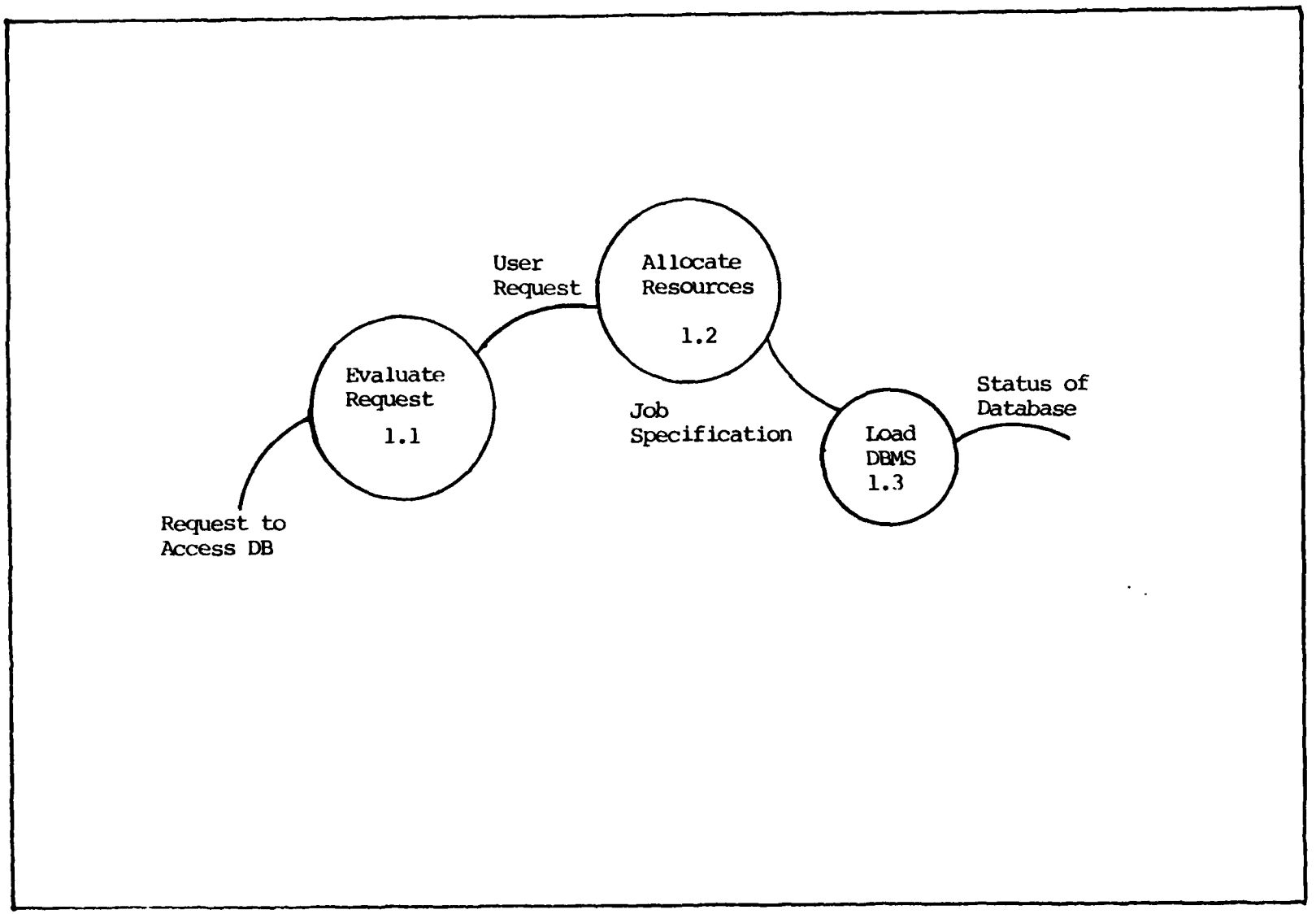


Figure M-3. OPENDB (Level 1.1)

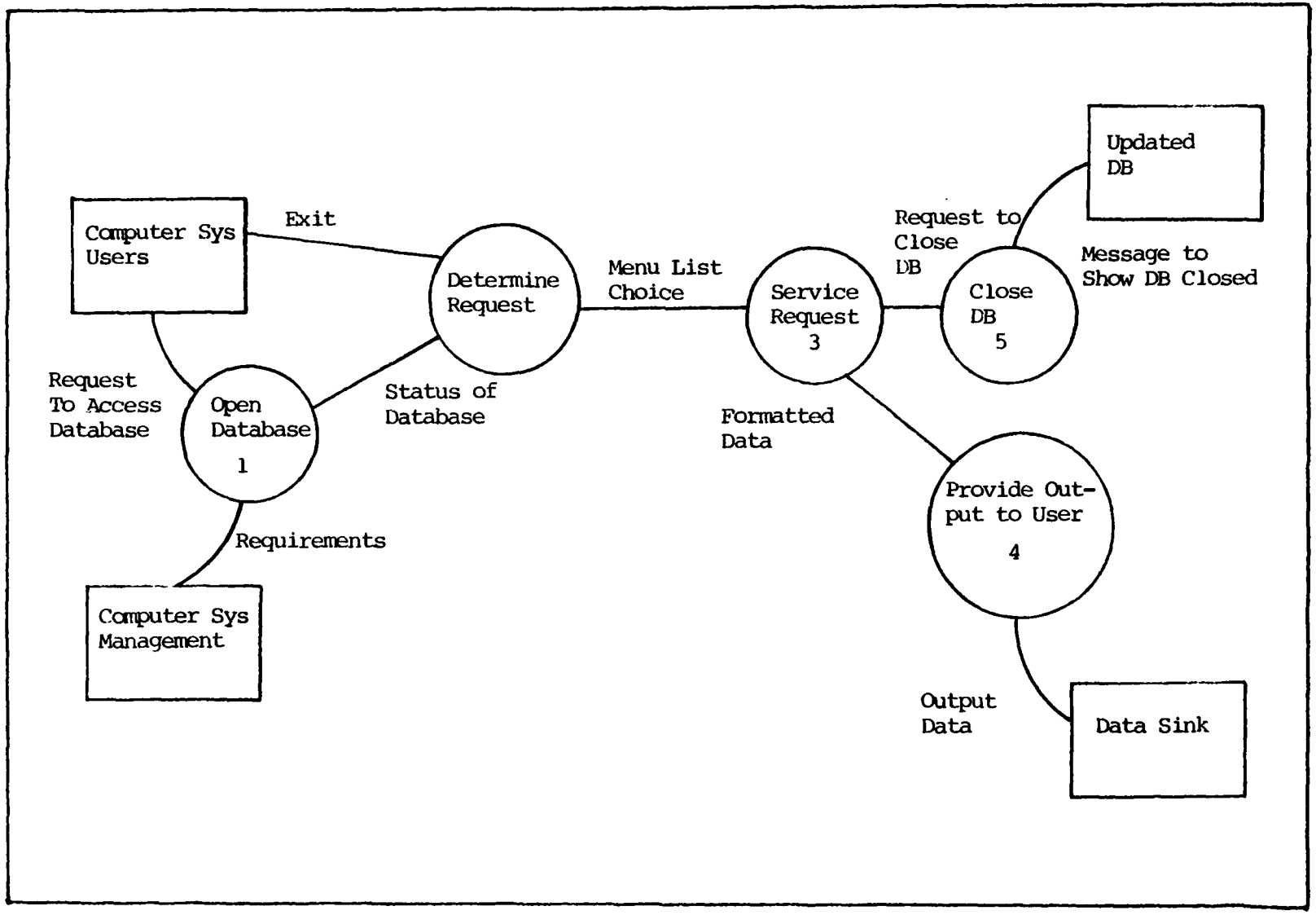


Figure M-2. AFIT Database Utilization (Level 1.0)

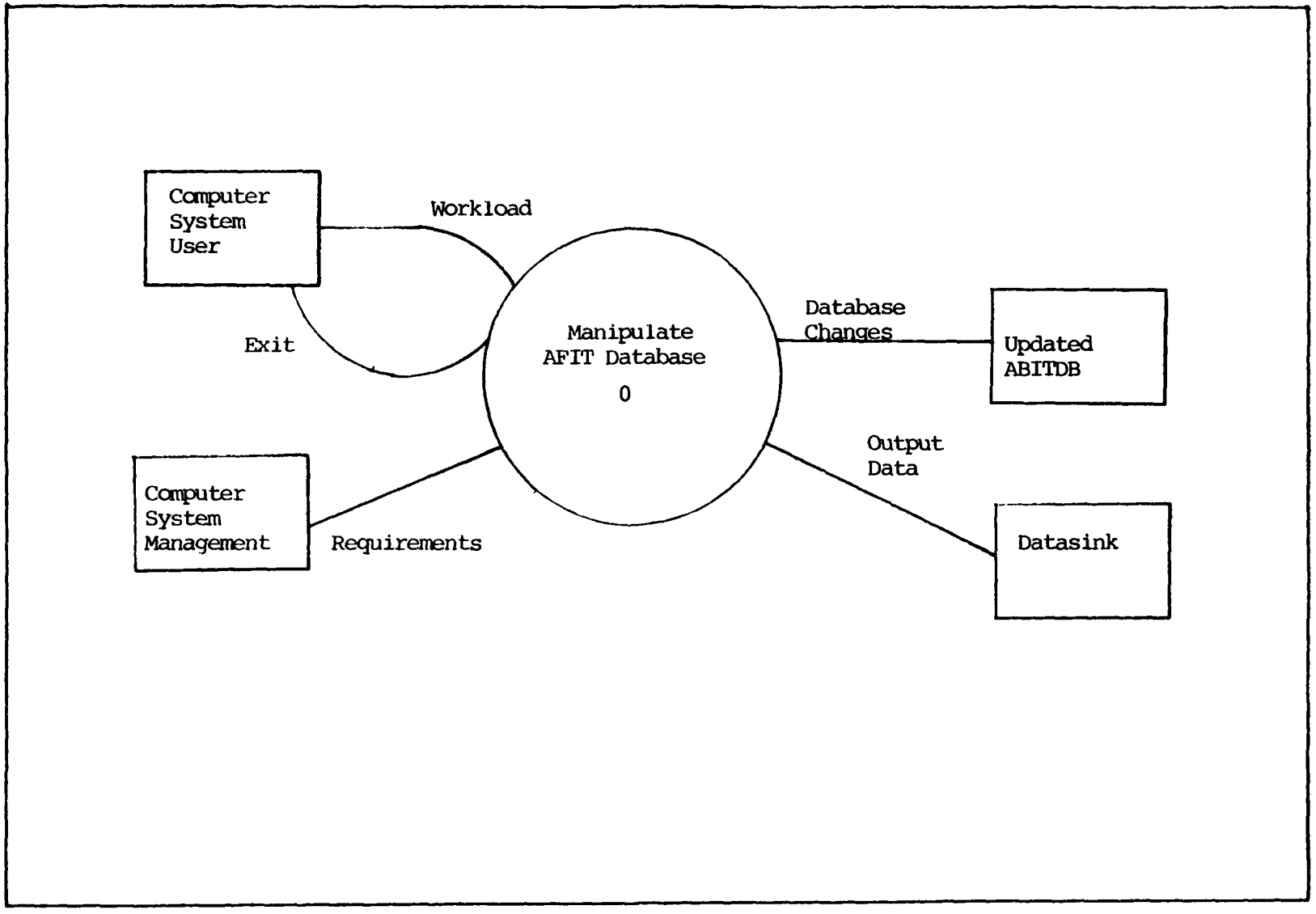
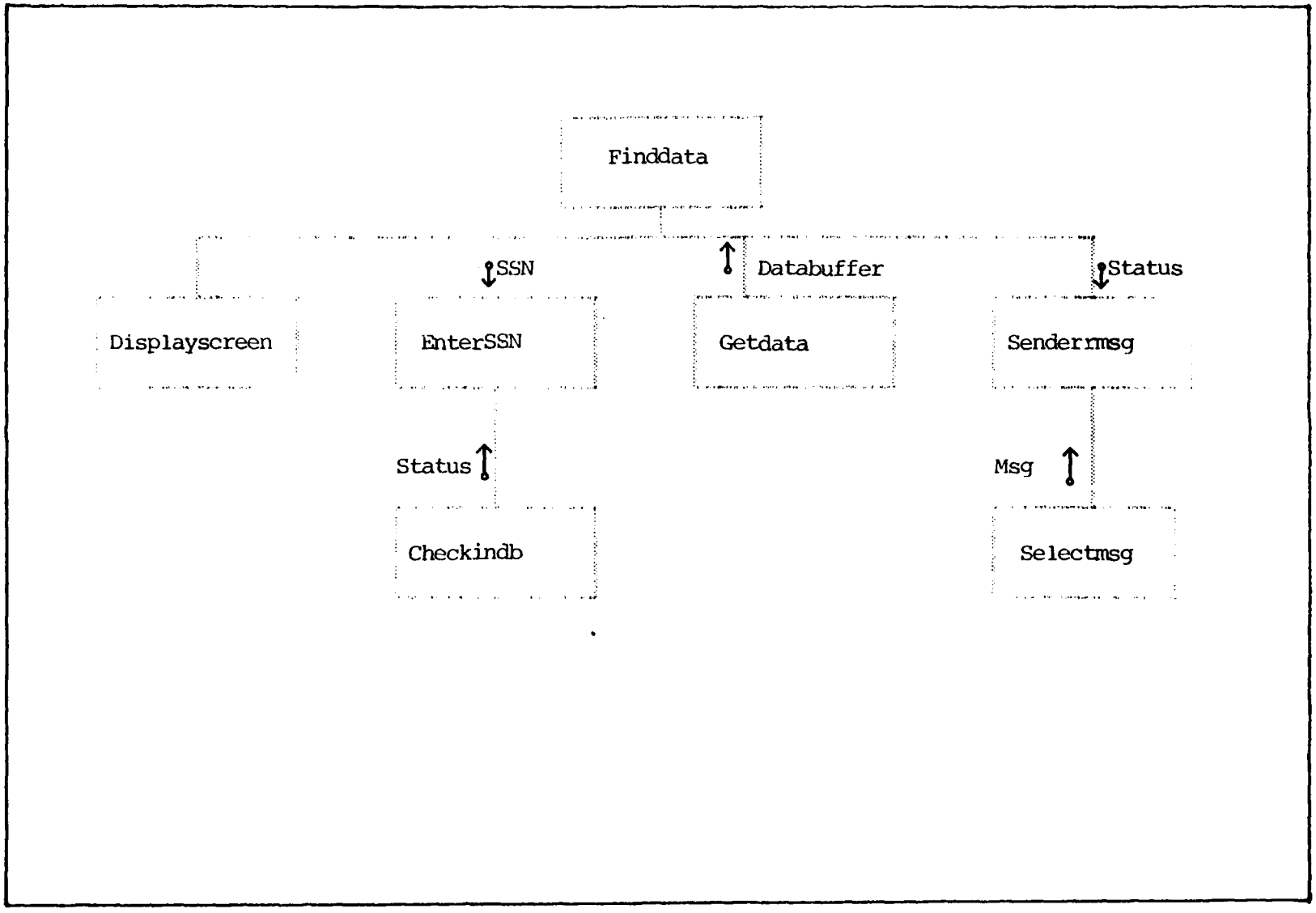
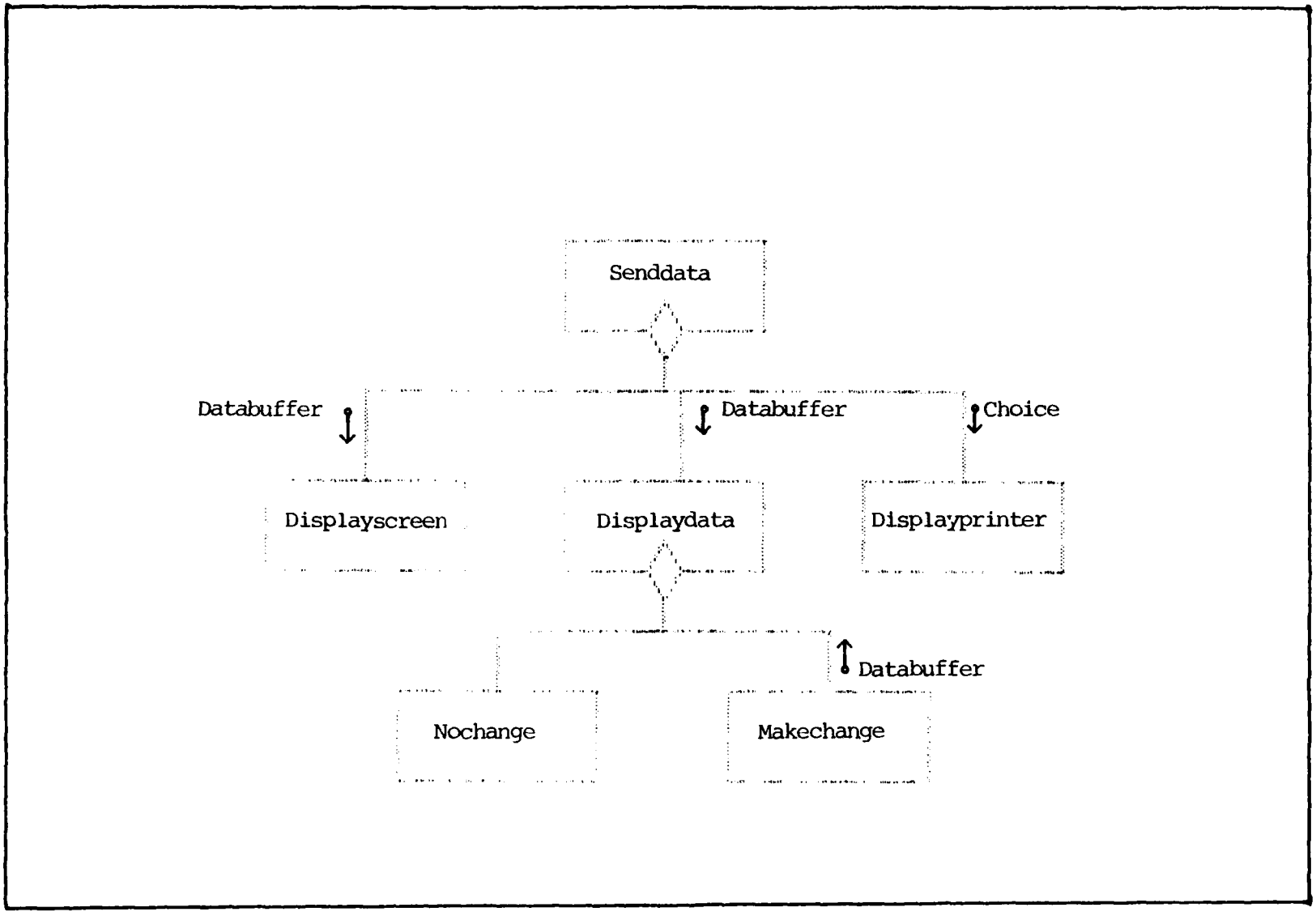


Figure M-1. AFIT Database Utilization (Level 0)

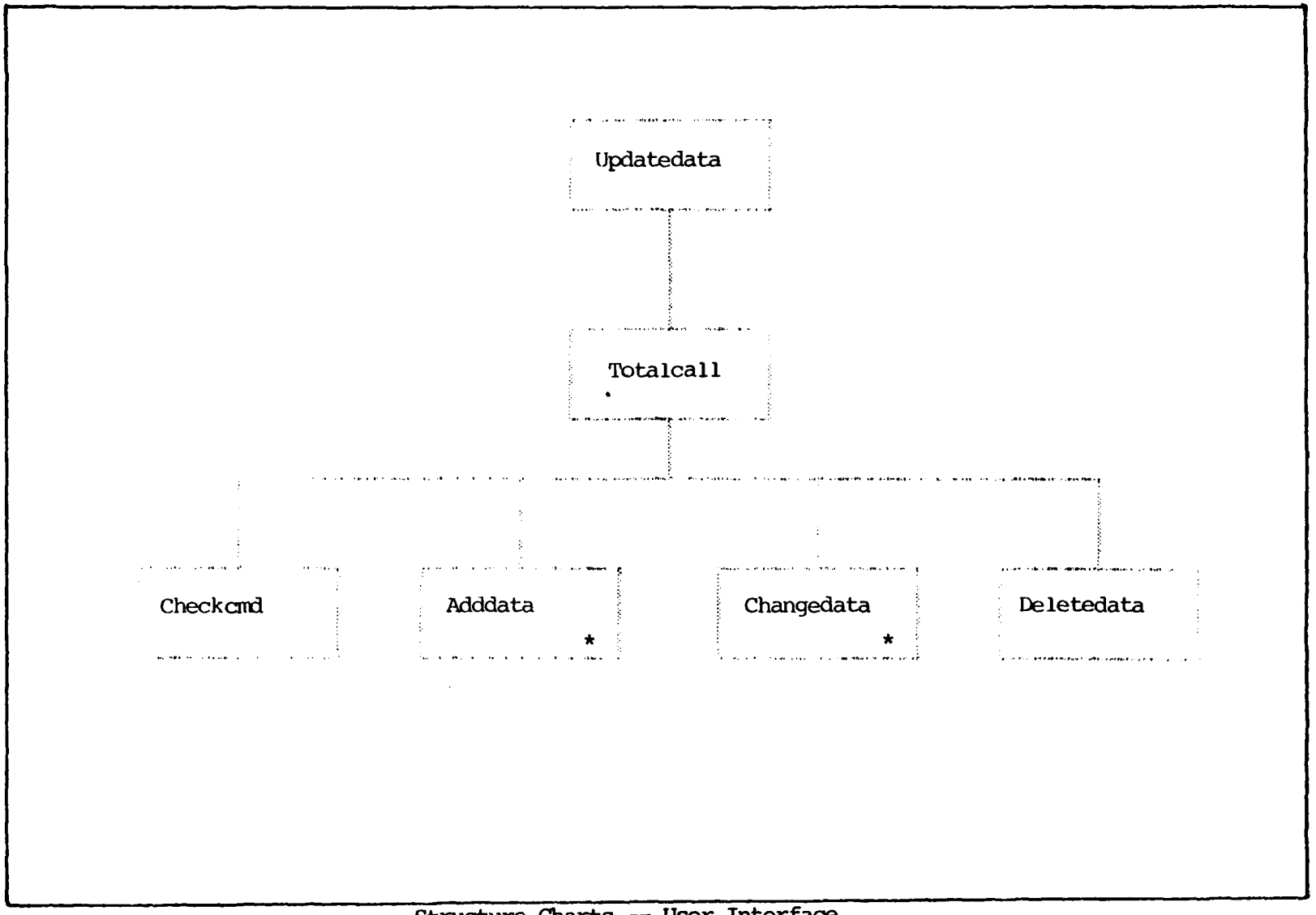
Data Flow Diagram Index



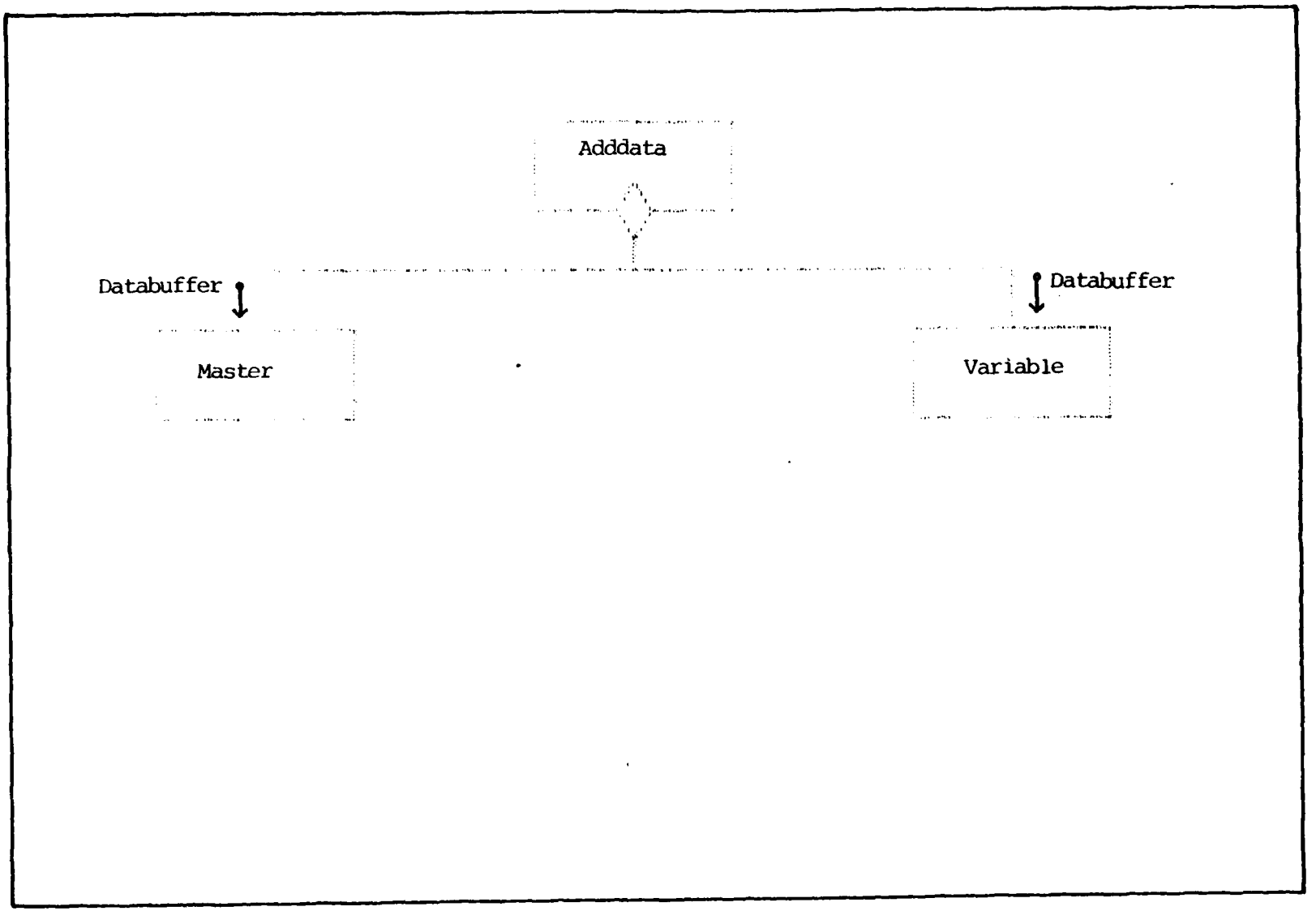
Structure Charts -- User Interface



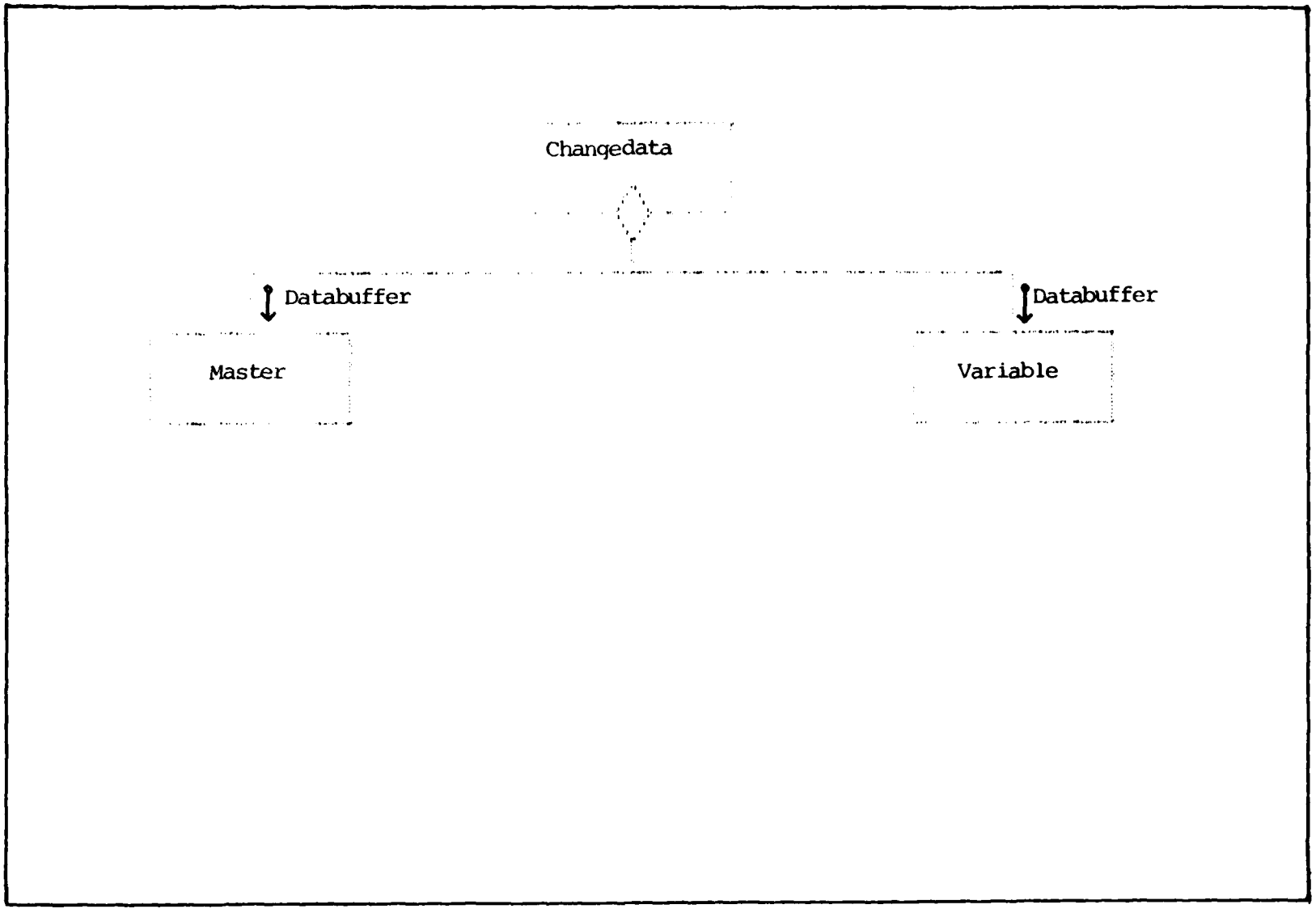
Structure Charts -- User Interface



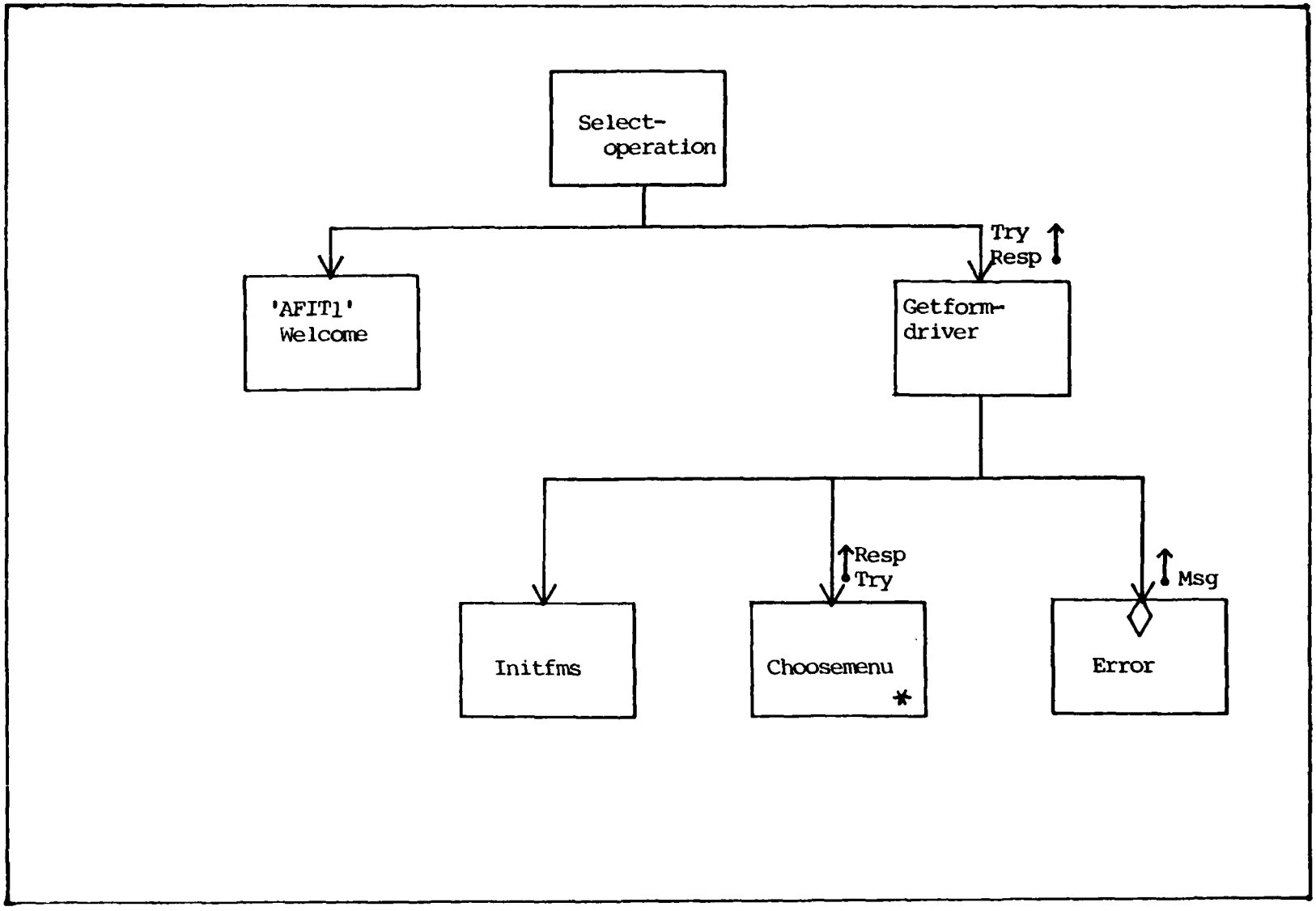
Structure Charts -- User Interface



Structure Charts -- User Interface



Structure Charts -- User Interface



Structure Charts -- User Interface

Appendix N

Proposed AFIT Database Administration

Many points of view have been asked for prior to and during the design of the database. Each interview produced a list of desires and expectations concerning the proposed database result (Appendices B and D). With this in mind it is not surprising that a requirement exists for someone to be responsible for not only what is contained within the database but also how data is safeguarded, changed and accessed. This requirement is filled by the Data Base Administrator (DBA).

Responsibility

The role of the DBA is most important not only to the database but also the organization. Changes to the database are expected. However, future changes can be done more rapidly, completely, and easily based on the documentation produced while building the database. The DBA should exert control to insure that any database changes come complete with proper documentation, and that all users and programmers adhere to accepted policies. These acts enhance the usefulness of changes affecting any part of the database and its subsequent use to the organization.

There is no universal definition of a DBA; it is unique to the enterprise (28:120). Ideally, the Database Administrator is the function within the enterprise that is created, organized, and staffed to manage the data resource. Having responsibility for the resource, the DBA must be granted the authority to control and manage the Database.

Inasmuch as there is effectively one organization involved in the control and use of the AFIT database, no discussion is offered as to the need to integrate the DBA with the other data generating or data using parts within the organization. This DBA automatically receives sufficient support from the Department Head within the Department of Electrical Engineering.

The DBA scope of responsibility is clearly not limited to the functional areas regarding data residing in a common file or those managed by a single DBMS. The integrated data are not the total resource and in many cases they are the smallest portion of the data. A definition of responsibility limited to integrated data is a clear invitation for user-generated and user-maintained files. The DBA must assume responsibility for, and control of all machine processable data, including the input and output files used to update the database and the report files generated from it. It must include those temporary files used by a one-time application, for the files that are created and maintained under a time-sharing system, and it must extend to include those files maintained by a service bureau or a facilities manager (28:122).

Functionality

To fulfill its responsibility to the enterprise as a resource manager, the DBA must lead the design of the database and any subsequent additions. The DBA must assume primary responsibility for all activities that are capable of altering the contents of the database and for those related activities that represent a significant investment in computer resources. The functions to be performed by the DBA may be divided into three areas: definition, existance, and utilization (28:124).

The definition aspects of the database are related to design, its reduction to a definition in the schema, the user's definition of the subschema, the data dictionary, and the narrative descriptions that are the basis of communication. Although these definition functions rely on the users, they are the sole responsibility of the DBA.

The physical functions associated with the existance of the database include initial load, accommodation of growth, quality assurance, protection, and restart and recovery. All must be assigned to the DBA in order to properly manage the resource effectively. This comes under the realm of security to some degree as well. One of the pressing concerns of a user is the assurance that 'his' data cannot be inadvertently damaged or accessed by someone else. This concern, bred in the era of tape, is based on the fact that if a program can access one of the records in an area of the database it can automatically access all of them. In the case where the DBMS discourages the use of multi record-types, the programmer must decide procedurally through coding which type of record he has accessed. To assure the user that an error in determining record type does not occur, it may be necessary to segregate records by type (28:76). This type of schema is the one we are familiar with in the TOTAL DBMS. Also, the ability to code variable data-sets within TOTAL can lend implicit security to the coding itself (19:4-17). Recovery can be concerned with validity, getting back to the correct state of the database, error correction routines, routines for taking snapshots or copies of the database (backups), and audit trail routines, to name but a few.

The third function assigned to the DBA concerns utilization of the database and efficiency considerations that effect database availability while it is operational. Such concerns include detection of procedures that cause resource contention through concurrent access protection achieved at the expense of availability regardless of the level at which access is controlled.

The tasks of restructuring and evaluating load density, program efficiency, and access patterns are normally accepted to be DBA functions. If the DBA is a resource manager, it must be possible to measure the database's performance. This is a task which has had two specific AFIT theses (23 and 25) completed already, and their results should be incorporated within the tasks required of the DBA. Proper application of these types of performance tools help to lend credibility to the DBA, for its job performance can aptly be measured in terms of the timely, accurate, consistent and complete data conducted during the normal usage of the database.

Also embedded in this operation can be the techniques employed by the users (or program authors). There is more than one way to write a program; consequently, there is the opportunity to do it either better or worse. Increasing the odds in favor of 'better' can also be a task the DBA would want to monitor. However, care must also be taken to insure that the efficiency improvements yield absolute improvements that are large enough to justify the effort required to attain them (28:126). To fulfill its responsibility for database accessability, the DBA must develop an active twofold program to improve the quality of applications that access the database to insure that designers and programmers are aware of the techniques.

The concerns of security continue to be an increasingly important matter. This will become more acute as the users expand to include most faculty and students within the Department. Since TOTAL can be configured to be multi-user (19:4-53), and the intent is for the database to serve the needs of the Department of Electrical Engineering, the requirement to allow students access to the database to look at and change their Ed Plans and course selections begs for a more comprehensive means to control access to particular data-sets, data-set items, and the use of particular database utilities.

Staffing

A DBA is not created to enlarge the data processing staff (if one exists) but to manage an existing resource - data. When the database talents are dispersed within the user groups, application specialists are distracted from their primary responsibility to cope with database problems.

The person selected to direct the activities of the DBA must provide the motivation for his staff and the functional users to use the resource effectively. He need not be an individual personally capable of filling each staff position. He should be technically qualified, be able to understand the subtleties of the database and be capable of managing technical specialists.

The staff must have at least one expert in the details of the DBMS software. In case of failure, this individual will play a most important role in reestablishing a properly functioning database. The DBMS specialist is required for the less dramatic aspects of the DBA to insure effective utilization of the DBMS for initial load, recovery, restructuring, and so forth (28:129).

The DBA staff must include persons with application programming and systems design experience. They must have experience with the use of the operating system, the hardware, and the utilities (like FMS) in order to perform their special functions.

Maintenance of the data dictionary does not require the same type of technician as the maintenance of software. Though not clerical, this job does require a different personality and temperament than is required of the software expert. The data dictionary can be maintained best by a person with an applications background who can appreciate the meaning of data and how they are used (28:130).

The quality assurance and audit functions require the stability of the applications specialist and the inquisitiveness of the technician. The way a database is used suggests those areas in which errors are likely to occur and where inconsistencies can be expected.

The AFIT/ENG DBA

Ideally, the DBA should consist of a director, a software maintenance expert, a person with a strong application background to interface with the users, a database techniques specialist, and the person who maintains the data dictionary, schema, and subschema and handles restarts. These are the primary functions. If possible, the DBA should contain positions that can be filled by user and data processing personnel. Of the many ways to improve the quality of the data processing and user staffs, one of the most effective is the active rotation of personnel among the different functional areas, including the DBA (28:126). This may or may not serve the same purpose within such a small organization like AFIT/ENG.

Taking into account the size and functions of the AFIT/ENG database, the DBA could properly be composed of one administrator, one software maintenance expert who also interfaces with the users and acts as the database techniques specialist, and one person who maintains the data dictionary, schema, and related resources. Additional theses concerning the AFIT/ENG database could draft the student undertaking the task and place him within the DBA as a DBMS or applications programs specialist.

The person selected to direct the activities of the DBA, called the Data Base Management Administrator (DBMA), will control all aspects of the utilization of the database, be responsible for any scheduled input or output matter and their format, designate the database schema design, and direct the inclusion/deletion of database application programs and subsequent record structures and database usage.

The busiest of the three, on a daily basis, regarding the database, will probably be the multi-faceted, software maintenance expert, the Data Base Technical Advisor (DBTA). His tasks will range from the actual database construction, security of the database, upkeep and expansion, write or oversee the writing and inclusion of application programs within the database, and act as the resident database expert for daily operations.

The Data Base Clerk (DBC) will maintain the data dictionary, schema, and current listings and modifications of the database and application programs. The Clerk will also maintain the configuration control and act under the direction of the DBTA and DBMA.

Appendix O

The Form Management System

An available utility on the existing hardware is the software package from Digital Equipment Corporation (DEC), the Form Management System (FMS). It is designed for ease of formulation and form simulation in order to collect and transmit data in an orderly manner. Forms are designed by typing them directly onto the terminal screen. Neither layout charts nor a special forms design language is required. FMS associates constant data with the form, not with the application program, resulting in simplified application program maintenance and increased application program flexibility. Forms may later be modified without the need to recompile the application program (as long as the form does not change application program specific areas). This utility supports any language supported by the VAX/VMS host computer (20:1-1).

The order of the data contained on the FMS screens is determined by grouping data items by category, and, to a lesser extent, by screen requirements. Each screen was created to present cohesive areas for data input and to make judicious use of limited space. Screen requirements pertain to the data collection precedence within the FMS (top-to-bottom and left-to-right). In an effort to show all personal data on the CRT screen at once without having to scroll through an area to see it all, the selection option was to forego the use of scroll areas. All personal data was still grouped by category. For an example, see Figure I-5. The most often accessed student data precedes the data less often changed to accommodate the anticipated changes required to be performed

by the user (secretary, student, etc). Thus, a student's rank (item 5) and service (item 7), precede the student's place of birth (item 22) and his previous command (item 28), which will not conceivably change while at AFIT.

The Form Editor

The Form Editor (FED) simplifies designing, modifying, and storing form descriptions for video display. The screen always shows the current state of the form while it is being edited. Keypad and keyboard functions provide ways to specify video display characteristics for constant text or fields that contain picture characters. To help operators, short, helpful explanations about individual fields may be included as well as separate help forms as required (20:1-2).

When designing forms, the programmer assigns form names and field names to reference data used (but not displayed) by the Form Driver when the form is used by (or within) an application program. Also, the desired operator response to information displayed or data to be entered on forms is controlled by the actual design of the form and the specific application requirements (20:1-2). An example is in FMS form STDT.FRM. When providing "MIL/CIV RANK" data, the form only accepts three characters within the field. The first character must be an alphabet character and the other two characters must be digits 0-9. Failure to provide the expected response for each character, rings the terminal bell and rejects the input. This is not to be confused with complete entry error checking. An expected response to this field is '04' (for MAJ rank) or 'G12' (for civilian grade). An accepted response could be 'w89'.

Creating or editing a form with the Form Editor is interactive and iterative. After creating a form, it can then be combined with others to form a library from which they can be retrieved and used by specific application programs. The most useful part of this utility is the ability to change a form format or look (such as using reverse video attributes for any part or all of the form) after the form is already part of an application program and have little or no impact upon the related application program (20:2-1).

Unfortunately, nothing is for free. The direct linking of the utility with an application program was not described or explained in the reference (17). Also, all applications were expected to be written in Pascal. However, no documentation for the interconnection or relating of this utility using the Pascal language was available. This caused time spent to determine how to relate the utility with Pascal, after first determining how to write and enable a working driver in Fortran. This drawback was accomplished at the expense of time which could have been utilized creating requested application programs.

The Form Utility

The Form Utility (FUT) allows the programmer to create versions of form descriptions that are suitable for hard-copy listings, to create and modify form libraries, and to list the names of forms contained in a form library. This also includes extracting and deleting form descriptions from form libraries, and combining form descriptions and form libraries into large form libraries (20:1-2).

The Form Driver

The Form Driver (FDV) is the heart of the usefulness of the utility. It is a set of subroutines that permit the application program to access forms created using the Form Editor. In an application that uses video images of forms on the terminal screen, using the Form Driver can reduce the programming effort by manipulating the screen, checking responses that an operator types, and displaying help messages and forms when the operator requests them (20:1-2).

Application programs access forms by issuing Form Driver calls that are embedded in the program and are written in the source language of the program. All Form Driver calls refer to specific forms and/or fields within forms by names assigned during the form editing process.

The Form Driver performs field and character validation for operator input based on the form definition (validation is based on picture validation characters and field attributes). The Form Driver also responds to operator 'Help' requests by displaying help text associated with the form and field being processed (20:4-2).

The Form Driver is by far the most complicated and time consuming of the major portions of this utility to initialize and get working. The provided manual is very helpful but it does not provide reference for actually using the utility in an application. The reference does provide explanations and a few written programs to use as guides. However, the sample program source code provided had numerous errors which prohibited it from compiling and running.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION [illegible]	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) [illegible]-20	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION [illegible] / 6b. OFFICE SYMBOL (If applicable) [illegible]	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) [illegible]	
12. PERSONAL AUTHOR(S) [illegible]	
13a. TYPE OF REPORT [illegible] / 13b. TIME COVERED FROM ____ TO ____	14. DATE OF REPORT (Yr., Mo., Day) [illegible] / 15. PAGE COUNT [illegible]
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	[illegible], Network Database, [illegible] Management System), Input/Output [illegible]
[illegible]	[illegible]		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

[illegible]

Approved for public release: IAW AFR 190-17.
LYNN E. [illegible]
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION [illegible]
22a. NAME OF RESPONSIBLE INDIVIDUAL [illegible], Department	22b. TELEPHONE NUMBER (Include Area Code) [illegible] / 22c. OFFICE SYMBOL [illegible]

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

This study undertook a complete revision and redesign of the AFIT/ENG database in an effort to expand existing capabilities and utilities. The Software Development Life Cycle approach was utilized throughout this project in order to provide a basis for sound software engineering principles and provide for a sound database development base. The initial compilation of data was obtained by interviewing a wide range of individuals to determine essential needed functions as well as requested data requirements. The results were combined to form the specific network master and variable data-sets for use with anticipated application programs for a greater database utility.

The expanded database exhibits the means to manipulate student and faculty data as well as provide for the inclusion of advanced data handling routines like Education Plan and Graduate Credit Record utilities. To insure standard data input and output as well as provide a more user friendly environment, a menu form display technique was incorporated within the expanded database driver. The resultant program was designed for modularity and expansion with little additional coding required to connect future database application routines.

END

FILMED

5-85

DTIC